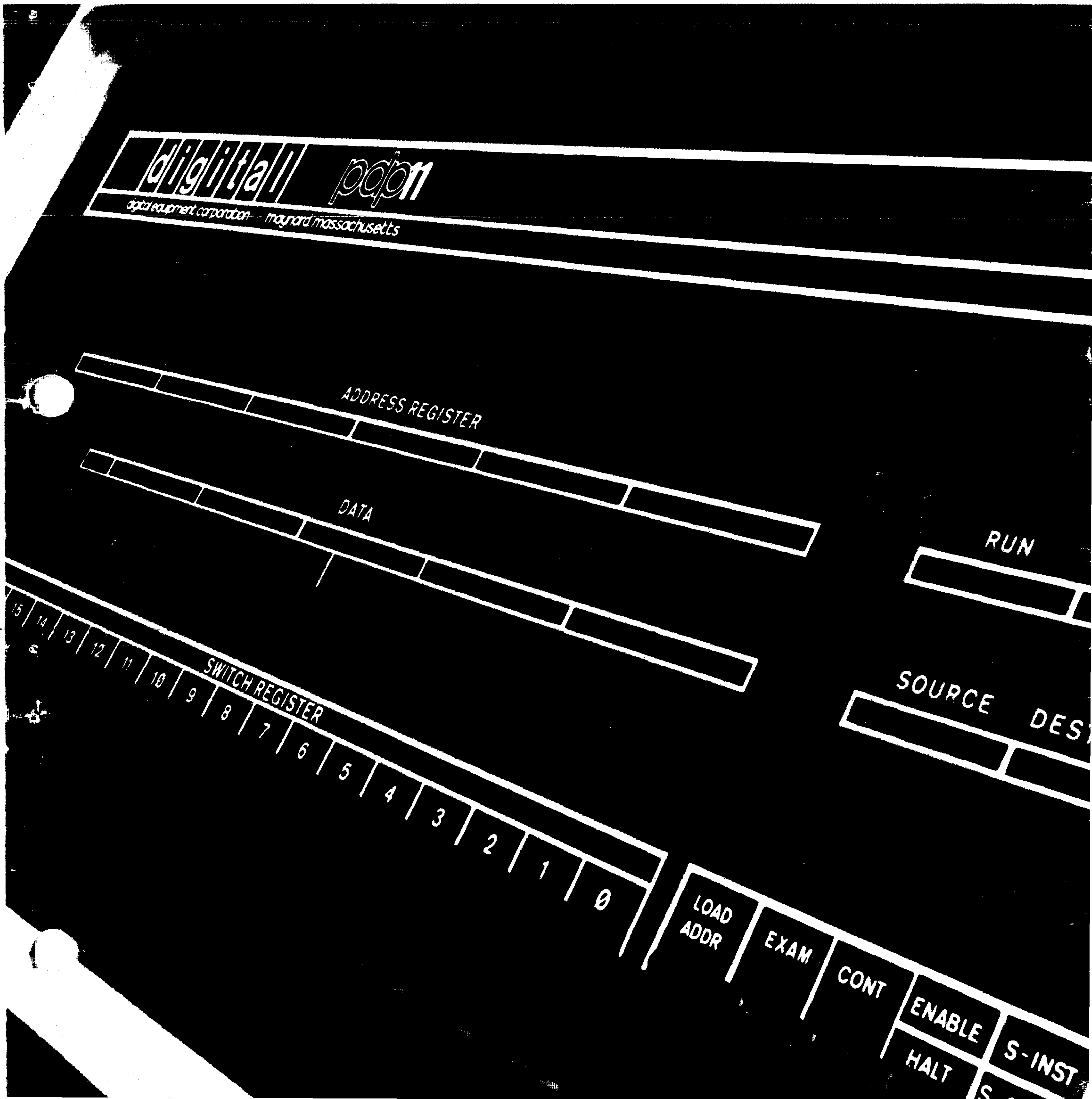
digital

pdp11

BATCH user's guide



DEC-11-OBUDA-A-D

B A T C H

U S E R 'S   G U I D E

Monitor Version No. VØØ6A

<u>SOFTWARE SUPPORT CATEGORY</u>

The software described in this document is supported by DEC under Category I , as defined on Page V of this document.

For additional copies, order No. DEC-11-OBUDA-A-D from DEC, Software Distribution Center, Bldg. 1-2, Main St., Maynard, Mass. 01754

pdp11 PROGRAM LIBRARY

First Printing, May, 1972

Your attention is invited to the last two pages of this document. The "How to Obtain Software Information" page tells you how to keep up-to-date with DEC's software. The "Reader's Comments" page, when filled in and mailed, is beneficial to both you and DEC; all comments received are acknowledged and are considered when documenting subsequent manuals.

Copyright © 1972 by Digital Equipment Corporation

Related documents:

DOS Monitor Programmer's Handbook, DEC-11-OMONA-A-D

FORTRAN IV Programmer's Manual, DEC-11-LFIVA-A-D

Link-11 Linker and Libr-11 Librarian Programmer's Manual, DEC-11-ZLDC-D

PIP File Utility Package Programmer's Manual, DEC-11-UPUPA-A-D

MACRO-11 Assembler Programmer's Manual, DEC-11-OMACA-A-D

Edit-11 Text Editor Programmer's Manual, DEC-11-EEDA-D

The following are registered trademarks of Digital Equipment Corporation:

DEC	PDP
DIGITAL (Logo)	COMTEX
UNIBUS	RSTS
DECtape	RSX

PREFACE

The Batch User's Guide describes the PDP-11 Batch operating system. The manual is organized into four chapters, augmented by five appendixes.

Chapter 1, "How to Use Batch" is directed primarily to FORTRAN programmers, who require information on how to prepare a job for Batch execution; and to the operator, who must know how to invoke Batch, and how to communicate with the system. Basic information is provided on the Batch Command Language; i.e., only those commands essential to executing a job are described ($JOB,$EXECUTE, $RUN, $FINISH). A brief glossary is also provided. Little or no knowledge of DEC software is required to understand and use Batch, as it is described in Chapter 1.

Chapter 2, "Batch Command Language", covers the entire complement of commands used in Batch. The reader of Chapter 2 will benefit from familiarity with the PDP-11 Disk Operating System.

Input/output is discussed in Chapter 3. Again, a knowledge of DOS is assumed.

Chapter 4 is directed to programmers who are responsible for writing, or modifying programs that are used in conjunction with Batch.

The appendixes provide useful information related to Batch; how to use the CR11 Card Reader, special punches for end-of-file and brackets, and, how to use the LP11 Line Printer.

NOTATION CONVENTIONS IN THIS MANUAL

Brackets [ ] - brackets are used to enclose optional elements; e.g., FI[NISH] - indicating that the letters NISH may be omitted at the user's discretion. The brackets are not part of the command.

Note

An exception to the use of brackets is in conjunction with the user identification code. The brackets are required, and must appear as part of the statement; refer to Chapter 1, Section 1.1.2.

Delta Δ - the Δ character is used to denote "space", as in "fileΔname," which means that "file" and "name" must be separated by a space.

Braces { } - braces are used to enclose two or more elements, from which a choice must be made.

For example

$$\left\{\begin{matrix}\Delta\\ ,\end{matrix}\right\}$$

means that either a space or a comma must appear in the position.

SOFTWARE SUPPORT CATEGORIES

Digital Equipment Corporation (DEC) makes available four categories of software. These categories reflect the types of support a customer may expect from DEC for a specified software product. DEC reserves the right to change the category of a software product at any time. The four categories are as follows:

CATEGORY I
Software Products Supported at no Charge

This classification includes current versions of monitors, programming languages, and support programs provided by DEC. DEC will provide installation (when applicable), advisory, and remedial support at no charge. These services are limited to original purchasers of DEC computer systems who have the requisite DEC equipment and software products.

At the option of DEC, a software product may be recategorized from Category I to Category II for a particular customer if the software product has been modified by the customer or a third party.

CATEGORY II
Software Products that Receive Support for a Fee

This category includes prior versions of Category I programs and all other programs available from DEC for which support is given. Programming assistance (additional support), as available, will be provided on these DEC programs and non-DEC programs when used in conjunction with these DEC programs and equipment supplied by DEC .

CATEGORY III
Pre-Release Software

DEC may elect to release certain software products to customers in order to facilitate final testing and/or customer familiarization. In this event, DEC will limit the use of such pre-release software to internal, non-competitive applications. Category III software is only supported by DEC where this support is consistent with evaluation of the software product. While DEC will be grateful for the reporting of any criticism and suggestions pertaining to a pre-release, there exists no commitment to respond to these reports.

CATEGORY IV
Non-Supported Software

This category includes all programs for which <u>no support</u> is given

CONTENTS

FIGURES

TABLES

CHAPTER 1

HOW TO USE BATCH

1.1 INTRODUCTION

This Chapter contains the basic information required to prepare a job for execution by the PDP-11 Batch operating system. The information is intended primarily for those users programming in FORTRAN, who wish to submit a job to be run under Batch. While the input medium is assumed to be the card reader in the following examples, Batch supports a variety of input devices: disk, DECtape, magnetic tape, or paper tape (refer to Table 1-2).

Batch includes a disk-resident Monitor, and a number of system programs (such as the FORTRAN Compiler). The Monitor controls execution of user jobs, by reading and interpreting batch command statements the user has placed in the input deck. Several jobs can be processed sequentially by Batch, each job set apart from its neighbors by delimiters that define its starting and ending points. The sample job shown below consists of a FORTRAN program to be compiled, linked,[1] and executed.

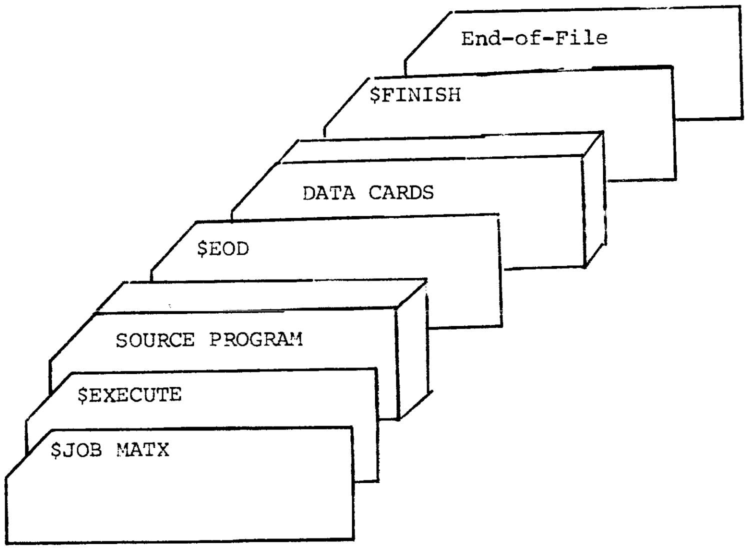


Figure 1-1 Sample Batch Job

[1]Linking is the process of producing an executable program from one or more separately compiled or assembled programs. The inputs to the Linker are called "object" modules; the executable output of the Linker is called the "load" module.

The function of each Batch command card shown in Figure 1-1 is defined as follows:

$JOB MATX	Identifies the start of the job, and assigns the job's name (MATX).
$EXECUTE	Calls the FORTRAN compiler, and causes the program to be compiled, linked, and executed.
$EOD	Acts as a logical data delimiter, separating the source program from the program's data cards.
$FINISH	Identifies the logical end of the job.
End-of file card	Identifies the physical end of the job. It contains a 12 11 Ø 1 6 7 8 9 in column 1; see Figure 1-2. (A "12" punch is the character & (Ø29 code) or a plus (+) (Ø26 code). An "11" punch is the minus (-) for either code.)

Certain commands (i.e., $JOB, $RUN, $CHANGE, $GET, $FINISH) are printed at the keyboard as they are read from the card reader. The card is echoed in its entirety, up to and including column 80. Therefore, to eliminate needless echoing of trailing blanks, put a "blank suppress" card at the front of the deck. This card has a punch of 12 11 Ø 7 8 9 in column 1.

Sequence numbers may be punched in columns 73-8Ø, as a means of keeping the deck in its proper order. The contents of columns 73-8Ø are not significant to Batch.

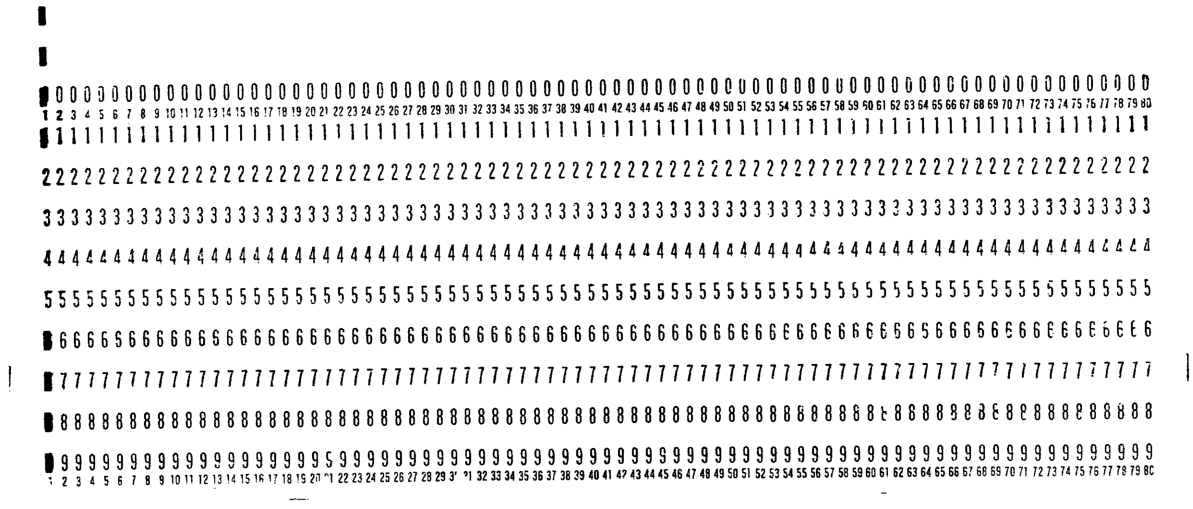


Figure 1-2 End-of-file card

When the $EXECUTE command is used, the Batch system determines the input and output specifications for the FORTRAN compiler. Thus, in the sample deck shown in Figure 1-1, the object program is output to the system device, the source listing is output to the line printer, and the source program is read in from the batch stream. The user can include input/output specifiers with the $EXECUTE statement, or he can use the $RUN command instead of $EXECUTE, followed by a command string. The command string consists of the input/output specifications, and is distinguished by having a # character in column 1. Figure 1-3 shows a typical command string. Note the "/GO" at the end of the # card - this is a switch that causes the FORTRAN program to be compiled, linked, and executed, just as though the $EXECUTE command had been used.

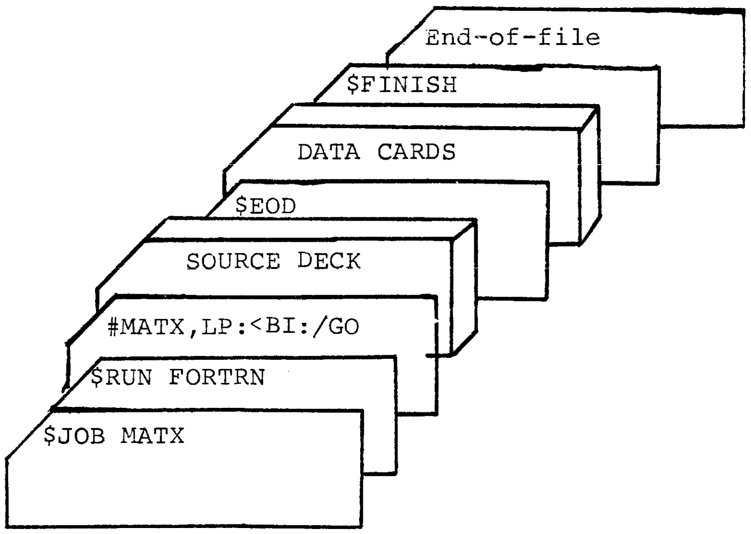


Figure 1-3. Use of /GO Switch

The "less than" symbol (<) separates the output specifications from the input specifications.

In Figure 1-3, the output is specified as:

MATX	-	object program (file name MATX; the extension .OBJ is assigned by default);
LP:	-	the line printer will be used to list the source program.

The input is:

BI:	-	the source program comes from the batch stream (BI);
/GO	-	the /GO switch causes the program to be compiled, linked, and executed.

The user also has the option of specifying each step of a job's processing. For example, he may wish to have the system generate a dump in the event of a fatal error in his program. To do so, he must include Batch Command Language cards at each step of the job.

As shown in Figure 1-4, the dump is specified on the $RUN MATX/DU card, by the switch, /DU. Optionally, this switch could have been used on the $RUN FORTRN card, in which case, the /GO switch could have been specified in the subsequent command string. The deck set-up would then duplicate that shown in Figure 1-3. A dump would be generated for a fatal error occurring during FORTRAN compilation, linking, or program execution. The dump linkage would be core-resident for the entire duration of the job.

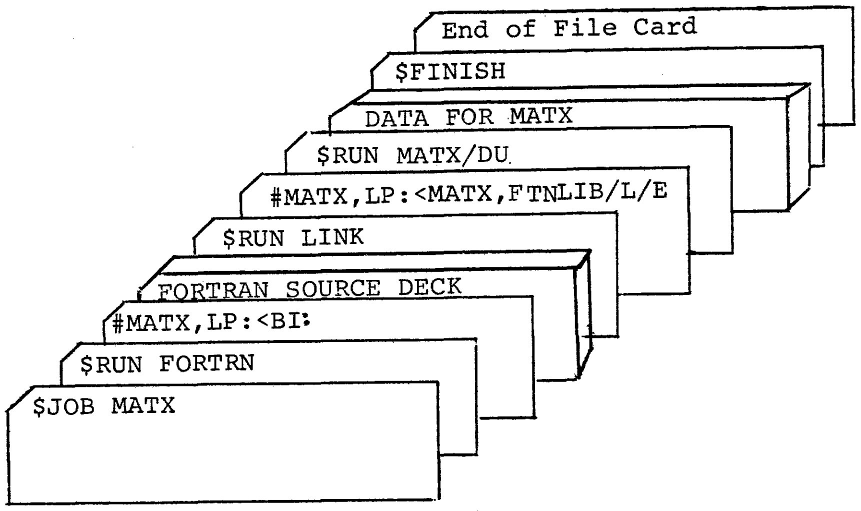


Figure 1-4 Batch Job Set-up with User-Specified Job Steps

Table 1-1 defines the function of each BCL card in the deck.

TABLE 1-1

Key to Card Deck in Figure 1-4

$JOB MATX	Defines the starting point of the job; and gives the job name, MATX.
$RUN FORTRN	Causes the FORTRAN Compiler to be loaded and executed.
#MATX,LP:<BI:	Command string defining the FORTRAN Compiler's input and output datasets. Note that the GO switch is omitted.
FORTRAN Source Deck	The cards comprising the source program, MATX.
$RUN LINK	Causes the Link-11 Linker to be loaded and executed.
#MATX,LP:<MATX/CC,FTNLIB/L/E (See note.)	Defines the Link-11 load module to be MATX; the Map output to be on the line printer; and the object modules to be MATX (the output of the FORTRAN Compiler) and FTNLIB, a Library routine (as indicated by the /L switch). The /E switch signifies "END" of Linker input.
$RUN MATX/DU	Causes the linked program, MATX, to be loaded and executed. The /DU is a "DUMP" switch, requesting that the program be dumped if an error occurs.
DATA FOR MATX	Data to be used during program execution.
$FINISH	The $FINISH card is the end-of-job delimiter; it designates the logical end of the job.
End-of-File	This card physically delimits the job. The operator must place one of these cards at the end of each job. As a further safeguard, the user should put one as the first card in his deck, to protect himself from a preceding job's erroneous execution. Eight consecutive end-of-file cards terminate the batch stream.

NOTE: The /CC switch (Concatenate) must be used, if the source deck comprises more than one main program or subroutine in a concatenated deck. The /CC tells the Linker that the object module contains concatenated modules. It is recommended that the /CC switch be used for all compiled object modules input to the Linker.

1.1.1 Terminology

batch stream	The sequence of statements (command or data) comprising the job(s) to be processed by Batch
batch stream device	The input device from which Batch reads the batch stream.
BCL	Batch Command Language. The set of statements used to direct the execution of a job.
dataset	A logical collection of data which is treated as a discrete entity by a program. A "dataset" can consist of data from a physical device, a file name (and optional extension); it may be a program, an object module, a load module, etc.
file	A physical set of data referred to by name. A file can be a card deck, or a name and collection of data residing on a disk, or DECtape.
job	The work to be done, delimited by a $JOB card and a $FINISH card. A job can consist of one or more programs.
Linker	A system program that takes one or more independently compiled programs as input and creates a single program that can be loaded and executed. The linked program is called a load module.
load module	The output of the Linker program, in load format.
log	The record of a job's processing; consisting of command statements processed, error messages, etc.
object module	The relocatable binary output of a compiler or assembler. Input to the Linker program.
overlay	A file, residing on a backup storage device (such as disk or DECtape), that comprises a portion of a program. Overlays are brought into core, as needed, thereby allowing a large program to run in a small core area, by "overlaying" an area in core (refer to Link-11 Linker and Libr-11 Librarian Programmer's Manual).

1.1.2 Job Definition

$JOB card identifies the start of a job, and permits the user to supply information pertinent to its execution. This card is formatted:

```
$JOB jobname[uic]/sw1/sw2[,log dataset]
```

where:

jobname	is specified by the user to assign a name to his job. This field consists of one or more letters or digits. The jobname could, for example, be the user's name (to help identify the destination of the log). Only the first six characters are used by the system. (The whole name is placed in the log, however.) If omitted, the job name is given a default value by Batch. (The first default job name is ØØØØØ1, the next is ØØØØØ2, etc. Defaults are assigned in numerical sequence. Each time a Batch session is started, the default sequence is reset.
[uic]	is the field in which the user identifies himself, by means of the "user identification code". This field is delimited by left and right brackets. The left bracket is multi-punched as either 12-8-2 (Ø29 code) or 11-8-5 (Ø26 code); the right bracket is either 11-8-2 (Ø29), or 12-8-5 (Ø26).[2] The format of the uic field is [nnn,nnn] where the nnn value to the left of the comma is an octal number identifying the user-group to which the user belongs, and the second nnn value identifies the particular user within that group. Thus, if the user has been assigned as user 27 within group 34, he would enter [34,27] for [uic]. If the uic is omitted, the default uic which is set by the system manager, is used. If the default uic is Ø, the job is not run.

[2] A punch in the 12 row is & (Ø29) or + (Ø26); an 11 punch is - for either. See Appendix B.

/<u>sw1</u>	is a switch used to set a limit on how long the job is permitted to run. It is formatted as: /TI:hh:mm where hh and mm are specified as decimal integers for hours and minutes respectively. If no time limit is provided, the BATCH command value is assumed. A value greater than that specified in the BATCH command is ignored (see 1.3.1.2). If only one value is supplied, it is assumed to be "minutes" (see sample job card, below).
/<u>sw2</u>	is the switch that allows the user to suppress the log, a record of the job's execution. If /NL is specified for /sw2, no log (record) is produced of the job control statements processed during the job's execution. If /NL is omitted, the user will get this record as part of his output. The device used for the log is specified in the BATCH command. (Refer to 1.3.1.2.)

<u>NOTE</u>

Switches can appear in either order; e.g., /NL/TI:Ø5

<u>log dataset</u>	is an optional entry, which specifies the dataset used for the log for this job. It overrides the log dataset specified in the BATCH command.

<u>SAMPLE JOB CARD</u>

```
$JOB MATX[34,27]/TI:15,DT1:LOG
```

<u>job name</u> - MATX

<u>user identification code</u> - user group 34, user 27

<u>time limit</u> - 15 minutes

<u>log</u> - put the log on DECtape 1, under file name LOG

1.1.3 <u>Compilation</u>

To compile a program, the user must place a $RUN card after the $JOB card. The $RUN card is a command to the Monitor to load and execute the compiler.
Format:

```
$RUN FORTRN
```

(For information on using MACRO-11, the PDP-11 assembler, refer to Section 1.3.3.)

Following the $RUN card, place a command string card defining the input to, and output from, the compiler.

Format:

```
#dev:obj-file,dev:list-file<dev:source-file
```

where:

#	(number symbol) occupies column 1;
dev	specifies the device to which the dataset is assigned (refer to Table 1-2);
obj-file	specifies the file name and extension of the object-code file;
list-file	specifies the file name and extension of the source program listing; and,
source-file	specifies the input file containing the source program to be compiled.

TABLE 1-2

Standard Peripheral Devices

Name	Mnemonic
Disk	DC DF DK
DECtape	DT
Line Printer	LP
Magtape	MT
Paper Tape Punch (High-speed)	PP
Paper Tape Reader (High-speed)	PR
Low-Speed Punch and Reader	PT
Keyboard	KB
Card Reader	CR
Batch Input	BI*
System Device	SY*

*This is a "pseudo-device". Refer to Section 1.1.3.1.

Command String Example

```
#MATX,LP:<BI:
```

The result of the above specification is:

- the object module is placed on the system device, under file name MATX.OBJ (OBJ is the default extension); the system device is the default output device.

- the listing of the source program is put on the line printer;

- the source program is read from the batch input device (BI), beginning with the next card in the stream.

1.1.3.1 Use of Pseudo-Device Specifiers (BI, SY)

By specifying BI as the input device, the user achieves device-independence; the source program is read from the batch input device, regardless of what the device may be. This feature of Batch permits the same control card to be used without concern for which device the batch stream may be read from.

To specify that the system device is to be used, when the actual device is not known, specify SY. The system will supply the correct device for SY.

1.1.3.2 FORTRAN Logical Units

The BI pseudo-device specifier is assigned to logical unit 8 in the FORTRAN device table. Logical units 1, 2, 3, and 7 refer to the SY specifier.

Examples:

```
READ (8,23) A, B, C     (read from BI)

WRITE (1,15)      (write to SY  - 2, 3, or 7 in place
                  of 1 also specifies SY)
```

The $ASSIGN command can be used to override the default values; e.g.:

```
$ASSIGN  BI:,4
```

Logical unit 4 is assigned to the batch stream dataset.

1.1.4 Linking

Object modules are linked into an executable program by the Link-11 program. Link-11 is invoked by the command:

```
$RUN LINK
```

This card must be inserted in the deck, following all source input to the compilation. The $RUN LINK card must be followed by a command string that specifies Link-11's input and output.

The command string has the following format:

```
#load module output,map output < object modules /E (or /GO)
```

The object modules are the inputs to Link-11 which are linked into the load module; the program that is to be executed. A load map may be obtained, which provides information on the load module (transfer address, high and low limits of the relocatable code, etc.) Details are provided in the _Link-11 Linker and Libr-11 Librarian_ manual.

Link Example

```
#MATX,LP:<MATX,FTNLIB/L/E
```

The two input modules specified are MATX (the Compiler's output), and FORLIB, which is a library module. (OBJ is the assumed extension of the input modules.) The load module is output to the system device, under the file name MATX.LDA.

1.1.4.1 _Creating Overlays in Batch Mode_

The overlay facility is described in the _Link-11 Linker and Libr-11 Librarian_ manual. To use this facility in Batch, the user supplies a sequence of command strings, defining the resident portion and each overlay, in the manner shown in the following example:

```
#RES,LP:<RES.OBJ/OV:2,FTNLIB[1,1]/L/E
#OVL1,LP:<OVL1.OBJ,FTNLIB[1,1]/L/E
#OVL2,LP:<OVL2.OBJ,FTNLIB[1,1]/L/E
```

The "OV:2" switch in the first command string specifies that the next two links are overlays. The user supplies a value of /OV corresponding to the number of overlays. Refer to the _Link-11 Linker and Libr-11 Librarian_ manual for details.

1.1.5 _Program Execution_

To execute a program, the user provides a $RUN statement, specifying the name of the program he wishes to be executed, and the input device on which the load module resides.

```
$RUN progname[uic]/switch
```

The switch that can be specified with the RUN command allows a dump to be taken should an error occur, providing the user with a debugging tool. In the example below, the core area for which the dump is requested is the area occupied by the user's program. Options are available to specify that all of core, or an area bounded by user-specified addresses, be dumped. (Refer to Chapter 2, "$RUN" command.)

Program Execution Example

```
$RUN MATX/DU
```

This example illustrates how a program named MATX would be executed (from the system device). In the event of an error, the program area would be dumped.

1.1.6 Summary of Job Preparation Steps

1. $JOB card, to denote the start of the job.
2. $RUN FORTRN (to invoke the compiler), or $EXECUTE. (Cards 3,5,6, not used with $EXECUTE.)
3. #MATX,LP:<BI: (the command string defining compiler's input and output). If /GO switch is specified, cards 5,6 not used.
4. Source cards; the FORTRAN source program.
5. $RUN LINK (to load and execute the Link-11 program).
6. #MATX,LP:<MATX,FORLIB/L/E (the command string defining the object modules, load module, and map to the Link-11 program).
7. $RUN MATX/DU (load and start the program MATX; dump the program area if an error occurs). Replace with $EOD if card 2=$EXECUTE; or /GO switch used with card 3.
8. Data cards.
9. $FINISH (to delimit the job, logically).
10. End-of-file card (physical end-of-job).
    This is only needed when the card reader is the batch stream device. This sequence of cards is repeated for each job to be processed during the Batch session.

1.2 BATCH COMMAND LANGUAGE

The user communicates with the Batch Monitor through Batch Command Language (BCL) statements. To prepare a FORTRAN job for execution, he must include statements to:

a. Define and delimit the job;
b. Effect compilation of source code;
c. Link object modules; and,
d. Execute the program.

Batch Command Language statements directed to the Monitor, must observe the following rules:

1. A dollar sign ($) must appear in column 1;

2. The statement identifier must immediately follow the $, starting in column 2.

3. The statement identifier is terminated by a comma or a space. Therefore, neither of these characters can appear as part of the identifier.

The $EXECUTE command, previously discussed, is one of the BCL commands referred to as "concise commands". They are called concise commands because they allow the user to invoke whole sequences of commonly-used functions with a single command, instead of two or more otherwise required. Concise commands are summarized in Table 1-3.

The full complement of Batch commands are described in Chapter 2.

TABLE 1-3

Batch Concise Commands

Name	Function
$CPY	Copies a file, or files, onto a specified output dataset.
$DEL	Deletes specified datasets.
$DIR	Obtains a directory listing.
$EX[ECUTE]	Causes a source program to be compiled (or assembled), linked, and run.
$FORTRN	Compiles a source program, producing an object module and a listing (a listing can be produced, with no object module, if desired).
$LINK	Links object modules into an executable load module, and generates a load map.
$LIST	Prints datasets on the line printer.
$MACRO	Assembles a source program into an object module, and produces a listing; or, if specified, produces only a listing.
$RNM	Renames a dataset.

1.2.1 $CPY

Function - Copy input dataset(s) to an output dataset.

Form - $CPYΔ input dataset(s) Δ TO Δ output dataset.

Input - At least one dataset must be specified; more than one may be specified Two or more input datasets are concatenated into one output dataset, if the output is a named file. Otherwise, they are separate. (Refer to the PIP manual.)

Output - One, and only one dataset, must be specified.

Examples

1.
```
$CPYΔDT2:*.OBJΔTOΔSY:
```

All files with extension OBJ, residing on DECtape unit 2, are copied to the system device.

2.
```
$CPYΔDK1:FIL.EXT[3,17]ΔTOΔMT2:NUFIL.EX1
```

The file, on RK11 disk unit 1, FIL.EXT (belonging to user 17 of group 3) is copied to magnetic tape unit 2. The name of the copy on magnetic tape is NUFIL.EX1.

1.2.2 $DEL

Function - Deletes specified datasets.

Form - $DELΔ dataset [,dataset2,...datasetn]

At least one dataset must be specified. If no device is specified, the system device is assumed. If a device is specified, it is assumed for following datasets that do not have device specifiers, until a device is specified.

Examples

1. `$DELΔA`

A file named A is deleted from the system device.

2. `$DELΔDT1:FILA.FTN,DK1:FLE.MAC,FLA.FTN`

FILA.FTN is deleted from DECtape unit 1; FLE.MAC and FLA.FTN are deleted from RK11 disk unit 1.

3. `$DELΔ*.MAC`

All files with extension MAC are deleted from the system device.

1.2.3 $DIR

Function - Obtain a directory listing.

Form - $DIR[[Δinput dataset(s)]ΔTOΔ[output dataset]]

Input - One or more input datasets can be specified. If omitted, the directory obtained is that of the user who is currently logged in. The default device is the system device.

Output - The default device is the keyboard.

Examples

1. `$DIR`

The current user's system device directory is printed at the keyboard.

2. `$DIRΔDF:`

The current user's RF11 disk directory is printed at the keyboard.

3. `$DIRΔDT1:,DK:[3,5]ΔTOΔLP:`

The current user's directory on DECtape unit 1, and user [3,5]'s directory on the RK11 disk, are printed at the line printer.

4. `$DIRΔTOΔLP:`

The current user's directory, on the system device, is printed at the line printer.

5. `$DIRΔ*.OBJΔTOΔLP:`

A directory listing of all files with extension OBJ that belong to the current user, and that reside on the system device, is printed on the line printer.

1.2.4 $EX[ECUTE]

Function - Compiles a source program, link the object module, and runs the resulting load module.

```
                                        binary
Form - $EX[ECUTE]Δ[source dataset][ΔTOΔ[dataset][,listing]]]
```

Input - The source program is assumed to be FORTRAN. The FORTRAN compiler is invoked to compile the source code specified in the "input dataset."

If the source dataset is not specified, the source program is assumed to follow the $EXECUTE statement in the batch stream. An $EOD statement is required to signal the end of source input and the beginning of data.

Output - If the binary dataset, and/or the listing dataset, are omitted, the object and load modules are temporary files on the system device, and the listing is produced at the line printer. Otherwise, the object and load modules are produced as specified; i.e., the load module assumes the object module's file name, with extension .LDA.

Examples

1. $EXECUTE

The FORTRAN compiler is loaded and run. The source program is read from the batch stream. The object module is output to the system device, linked into a load module, and run. The listing is printed at the line printer.

2. $EXECUTEΔPROG

The FORTRAN compiler is loaded and run, to compile the source program, PROG, from the system device. Linking and execution follow.

3. $EXECUTE DT1:ABC.CBAΔTOΔDK:ABC,LP:

The FORTRAN compiler is loaded and run. An object module (ABC.OBJ), and load module (ABC.LDA) are produced and placed on the RK11 disk. The load module is run. The listing is produced at the line Printer as requested.

1.2.5 $FORTRN

Function - Load and run the FORTRAN compiler, to compile source input dataset (source) and produce a binary output dataset and listing.

Form - $FORTRN[△[input dataset]△TO△[object dataset][,list]]

Input - If omitted, the source program is assumed to follow immediately in the batch stream.

Output - The object dataset, if omitted, goes to the system device, with the file name specified in the input dataset. If no input file name was given, the object module assumes the job name. The extension is .OBJ. The listing goes to the line printer, if defaulted.

Examples

1. $FORTRN△ABC△TO△XYZ,DT1:SRC

The source program, ABC, is read from the system device, and compiled, producing an object module. The object module is output to the system device, with the name XYZ.OBJ. The listing dataset goes to DECtape unit 1, with the name SRC.LST (LST is the default extension).

2. $FORTRN△DK:ABC△TO△SY:ABC

The source program ABC.FTN is read from the RK11 disk, and compiled; the object module goes to the system device, under the ABC.OBJ; the listing is produced at the line printer, by default.

3. $FORTRN△ABC

The source program, ABC, is read from the system device. The object module ABC.OBJ goes to the system device by default, and the listing is defaulted to the line printer.

4. $FORTRN

The source program is read, immediately following in the batch stream (i.e., BI: is assumed). The binary object module is put on the system device with the listing at the line printer, both by default.

5. $FORTRN△A△TO△,LP:

The effect is to generate a listing, at the line printer, but no binary dataset.

1.2.6 $LINK

Function - Invokes the Link-11 program, to link input datasets (object modules) into an output load module, and produce a load map.

Form - $LINKΔinput dataset(s)[ΔTOΔ[load module][,load map]]

Input - At least one input dataset is required; more than one can be specified. File characteristics, such as concatenated object modules (/CC) or library modules (/L), must be defined by the user (refer to Link-11 Linker and Libr-11 Librarian manual).

Output - Same as $FORTRN, except that the load module extension is .LDA, rather than .OBJ.

Examples

1.
```
$LINKΔA,FTNLIB/L
```

The object module, A, is linked with library routines from FORLIB, producing a load module (A.LDA) and a load map. The input comes from the system device (by default); while the load module and load map are output to the system device and the line printer, respectively, by default.

2.
```
$LINKΔDF:A,DT1:FTNLIB/LΔTOΔSY:ABC,LP:
```

The object module, A.OBJ is input from the RF11 disk, and linked with routines from FORLIB.OBJ, input from DECtape unit 1. The load module, ABC.LDA, is output to the system device; the load map is produced at the line printer

1.2.7 $LIST

Function - Print datasets on the line printer.

Form - $LIST[Δdataset1,...,datasetn]

Input - If no datasets are specified, the dataset immediately following in the batch stream is printed on the line printer.

Output- Line printer only. Never specified.

Examples

1. $LISTΔDK:A.FTN,B.FTN,DT1:Z.FTN

   Three datasets are printed at the line printer; DK:A.FTN, DK:B.FTN, and DT1:Z.FTN (DK: carries as the device specifier, until a different device is specified. If no device is specified, SY: is assumed.)

2. $LIST

   The dataset immediately following in the batch stream is printed.

1.2.8 $MACRO

Function - Assembles source input, producing an object module and a listing as output.

Form - $MACROΔinput dataset(s) {ΔTOΔ[object dataset][,listing dataset]}

Input - At least one input dataset is required; two or more can be specified. The MACRO-11 assembler assembles multiple input datasets together, creating a single object module.

NOTE

Input to the MACRO-11 assembler must be from a mass storage device. Source programs on punched cards must be copied to disk, DECtape, etc., prior to invoking the assembler.

Output - Same as that produced for $FORTRN; i.e., the object module defaults to the system device, and the listing defaults to the line printer.

Examples

1. $MACROΔA.MAC

   The source program, A.MAC, is assembled. The object module, A.OBJ, is output to the system device, by default. The listing is produced at the line printer, also by default.

2. $MACROΔAΔTOΔDT1:Z,DK:A

   The source program, A, is assembled, to produce object module, Z.OBJ, output to DECtape unit 1. The listing is placed on the RK11 disk, as A.LST.

3. $MACROΔA.MACΔTOΔ,LP:

   The result of this form of $MACRO is a listing on the line printer. No object module is produced.

1.2.9 <u>$RNM</u>

<u>Function</u> - $RNM renames an input file as specified in the output dataset.

<u>Form</u> - $RNMΔ"old name"ΔTOΔ"new name"

<u>Input, output</u> -Both are required. They must both be on the same physical device. If omitted, the default is the system device.

<u>Examples</u>

1. $RNMΔDT1:ABCΔTOΔDT1:XYZ

   The file, ABC, is renamed XYZ.

2. $RNMΔUNOΔTOΔDUE

   A file on the system device (UNO) is renamed DUE.

1.3 OPERATING PROCEDURES

1.3.1 Getting Batch On the Air

Two basic procedures are involved in getting started with Batch. The first procedure, loading the Monitor, is accomplished through the console. The second procedure, entering Batch mode, is done via the keyboard.

1.3.1.1 Loading the Monitor

Figure 1-5 shows the console of a PDP-11/20.

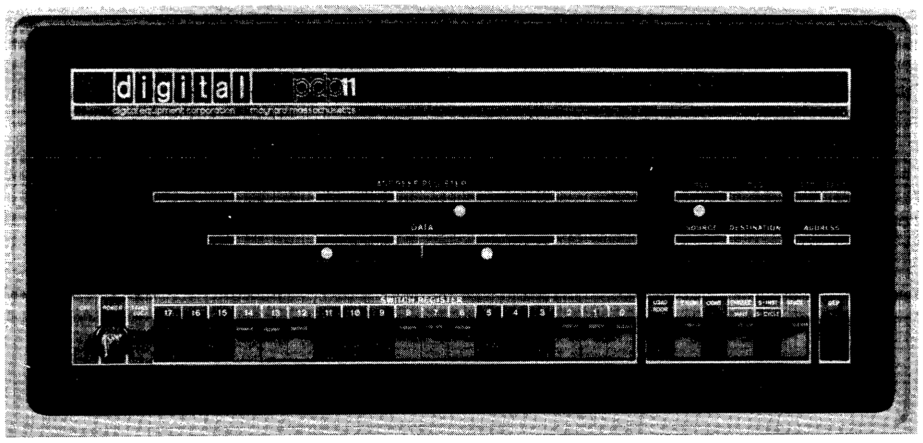
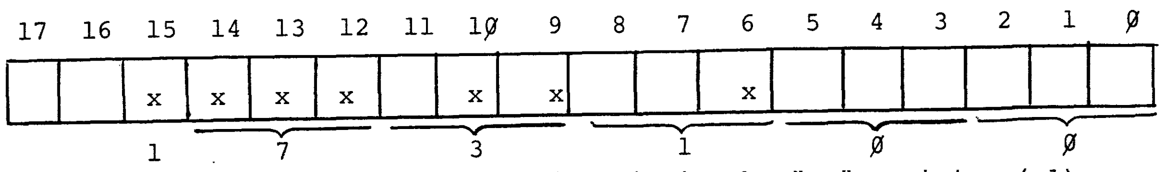

digital pdp11
digital equipment corporation maynard massachusetts
DATA
SOURCE DESTINATION ADDRESS
SWITCH REGISTER

Figure 1-5 PDP-11/2Ø Console

Proceed as follows:

1. Turn WRITE ENABLE off (for RK11 disk only).
2. Press HALT.
3. Set the Switch Register to $1731ØØ_8$; (see Figure 1-6).
4. Depress LOAD ADDRESS.



"x" indicates that the switch is in the "up" position (=1)

Figure 1-6 Console Switch

5. Set the Switch Register to the address of the word count register for the disk on which the Monitor resides (for RF11, normally 177462_8 for RK11, 177406_8; for RC11, $17745Ø_8$). See Figure 1-7 a, b, and c, respectively.

6. Turn ENABLE/HALT switch to ENABLE.

7. Press START.

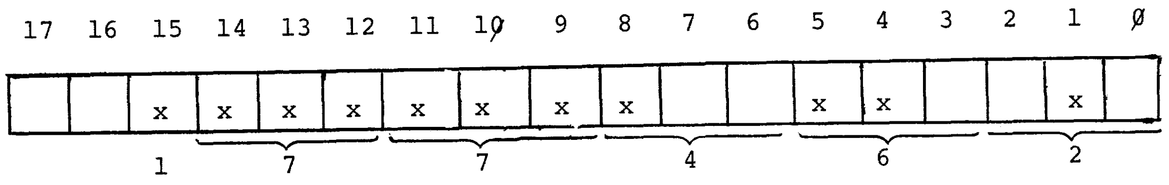


Figure 1-7a Set for RF11 Disk

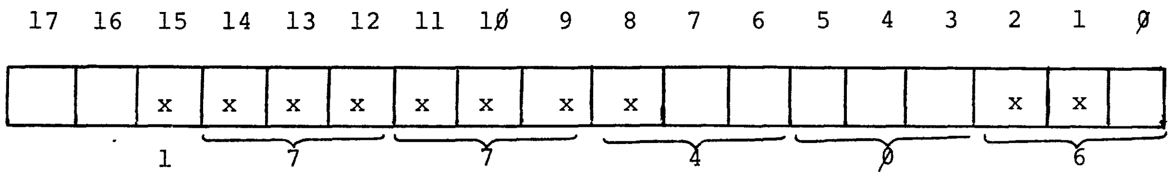


Figure 1-7b Set for RK11 Disk

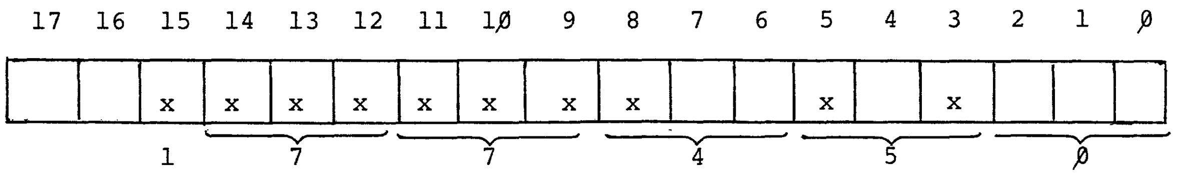


Figure 1-7c Set for RC11 Disk

x = this switch is in the "UP" position (equals 1).

Figure 1-7 Switch Settings for Word Count Register

When the Monitor has been loaded into core, it responds by printing:

```
MONITOR Vxxxx
$
```

at the teleprinter. Vxxxx identifies the version number of the Monitor currently in core, e.g., VØØ6A. Having identified itself, the Monitor issues a line feed, carriage return, and prints $ to indicate its readiness to accept an operator command.

Enter the date and time at the keyboard, using the DATE and TIME commands. For example:

```
DA: 24-MAR-72
TI: Ø9:15
```

Invoke PIP, and enter a UIC of [1,2], as:

```
$RUN PIP
   (PIP responds)
#[1,2]/EN
#↑C
.KI
```

1.3.1.2 Entering Batch Mode

When the Monitor responds by printing $, the appropriate response to invoke Batch mode is to type, immediately after the $, and on the same line:

```
BA dataset1/switch(es),dataset2
```

where:

dataset1	is the batch stream dataset
dataset2	is the default log dataset[3]

The switches that may be specified are the "time-limit" switch and the "no-echo" switch. The time-limit switch governs the maximum duration of a job. It is specified as

```
/TI:hh:mm
```

hh and mm are specified as decimal digits. If both are included, hh equals hours, and mm equals minutes. If only one value is given, it is assumed to be minutes.

The no-echo switch suppresses printing of commands at the keyboard. It is specified as /NE. If this switch is included, only the $JOB command is echoed at the keyboard, and the start and finish times of the job.

[3] Output related to syntax errors etc., is printed on the teleprinter, if the log dataset is omitted.

A NO FILE message at the keyboard indicates that the batch stream could not be found. The system searches first the current user's area; if it fails to find the batch stream there, it searches the system area. If the batch stream is not found there, the NO FILE message results.

The time-limit switch given with this BATCH command takes precedence over any specified in a $JOB command.

Sample BATCH Command

```
        BA CR:/TI:2Ø,LP:
```

The dataset specifiers in the example designate the batch stream device to be the card reader, with the log produced on the line printer. The default time limit is set to 20 minutes.

1.3.2 Operator-System Communication

Batch provides several ways for the operator and the system to communicate with each other: the Monitor may print information regarding the status of a job (e.g., error messages); a user program may request operator action; or, the operator may wish to exercise control of system operation, or respond to a system request.

1.3.2.1 Error Messages

The actions taken as the result of an error in Batch mode are described below. (The messages produced are summarized in Appendix F of the DOS Monitor Programmer's Handbook. Certain Fatal error messages pertinent to Batch are listed below in Table 1-4.)

The difference between Batch error processing and DOS error processing is mainly in the way these messages are presented to the user, since different classes of errors are handled according to their type.

Action messages (Annn) are printed on the teleprinter, but do not appear in the log returned to the user. The system suspends operation until the operator responds at the keyboard. Other classes of message (I, S, W, and F) are printed at the teleprinter, and in the log (unless the user has suppressed the log).

TABLE 1-4

BATCH MESSAGES

Error Code	Additional Information	Meaning
AØØ1	User Call Address	Disk address error. Type KILL to abort job, then CTRL/C to start next job.
AØØ2	Device in Radix-5Ø format	Device not ready. Make device ready and type CO. See Appendix E for device name.
AØØ3	Link Block Address	The Link Block contains either an illegal device code or no device code at all. The recommended procedure is to terminate the job (KI).
AØØ4	User Call Address	DECtape error. Try adjusting the tape; type CO to continue.
AØØ5	Pause Number	A PAUSE was encountered in a FORTRAN program. Type CO to continue.
AØØ6	Correct Module Name	Loading paper tape out of order on Pass 2 of Linker. Load correct module and type CO to continue.
AØØ7	Call Address	Magtape. The name of the output file being created is the same as that of an existing file. Type KI to terminate the job.
AØ1Ø	Ø	Magtape. A parity error occurred when trying to open a file. Type CO to continue searching. If the file being sought has a parity error, it cannot be found.
AØ11	Ø = time 1 = date	Date or time is invalid. Enter correct values via DATE or TIME command and type CO.
AØ43	Disk Pack Block Number	This is the block that is bad; issued by the RK11 pack initializer to provide a list of bad blocks and to permit job termination if too many are bad. Type CO to continue, if number of bad blocks thus far is tolerable.
AØ5Ø	Ø	Indicates that a $WAIT statement was processed from the batch stream. Type CO to resume program operation. eration
FØ5Ø	PC	Illegal I/O request to the batch stream.
FØ51	PC	Too many consecutive EOF's or binary errors from the batch stream.

TABLE 1-4 (cont'd)

Error Code	Additional Information	Meaning
FØ52	PC	Illegal open to one of the batch datasets.
FØ53	PC	Illegal request to the batch stream bypass EMT.

Note: Radix-50 device codes are summarized in Appendix E.

If an S (System Program) error occurs, subsequent input from the batch stream is ignored, until another command string is detected, or a Monitor command is read.

A Fatal (F) error causes the job to be aborted. If the user has specified "dump-on-error," he is given an octal listing of the contents of the area he specified to be dumped.

1.3.2.2 Messages to the Operator

Messages can be sent to the operator from the batch stream, by means of the $ME command, which is formatted:

$ME $\{{}^{,}_{\Delta}\}$text

The $ME command adheres to the syntax conventions used by the other Batch commands; i.e., the $ must be in column 1. The command itself is separated from the message by either a comma or a space, as indicated by $\{{}^{,}_{\Delta}\}$. (The braces are not part of the command syntax, but merely indicate that one of the enclosed characters must be used in this position.)

The program will continue execution, following the issuance of $ME text, unless the user issues a $WAIT command; in this case, the operator is required to type in CO to effect resumption of the program.

EXAMPLE

Entries supplied by User, in batch stream (assume job name to be MATX):

```
$ME MOUNT TAPE XYZ ON UNIT 1
$ME DO NOT WRITE ENABLE
$WAIT
```

Response Printed at Keyboard (as seen by the operator):

```
MATX   :MOUNT TAPE XYZ ON UNIT 1
MATX   :DO NOT WRITE ENABLED
AØ5Ø   ØØØØØØ     (action message indicating that
                  $WAIT statement is in effect)
$
```

The operator types CO on the $ line to resume the program, after the action has been taken.

1.3.2.3 Operator Commands

The operator notifies the Monitor of his intention to type in a command by pressing the CRTL and C keys simultaneously. (This action is indicated as CTRL/C.) The Monitor responds by printing a period at the start of the next line. The operator then types in the appropriate command.

To abort the current job, the operator/Monitor message sequence is:

```
CTRL/C
.KI↵     (the period (.) is printed at the keyboard in
         response to CTRL/C)↵ stands for "carriage return"
```

To terminate the batch stream, the operator types

```
CTRL/C
.TE↵
```

following which, the system leaves batch mode, returning the Monitor to keyboard mode.

1.3.2.4 Commands Printed at the Keyboard

The Batch system prints some commands at the keyboard, to help the operator monitor a job's progress. For example,

```
        $RUN FORTRN
```

is printed at the keyboard when this card is read. (It is also output to the job log.) $JOB, $GET, $CHANGE and $FINISH are also printed. (If the /NE switch is used, only $JOB is printed.)

1.3.3 MACRO-11 Assembly Procedure

The MACRO-11 Assembler makes two passes over the source file. To assemble a MACRO-11 program, the user must first transcribe the program to a mass storage device, by means of the PIP system program. This is done as shown in the example in Figure 1-8.

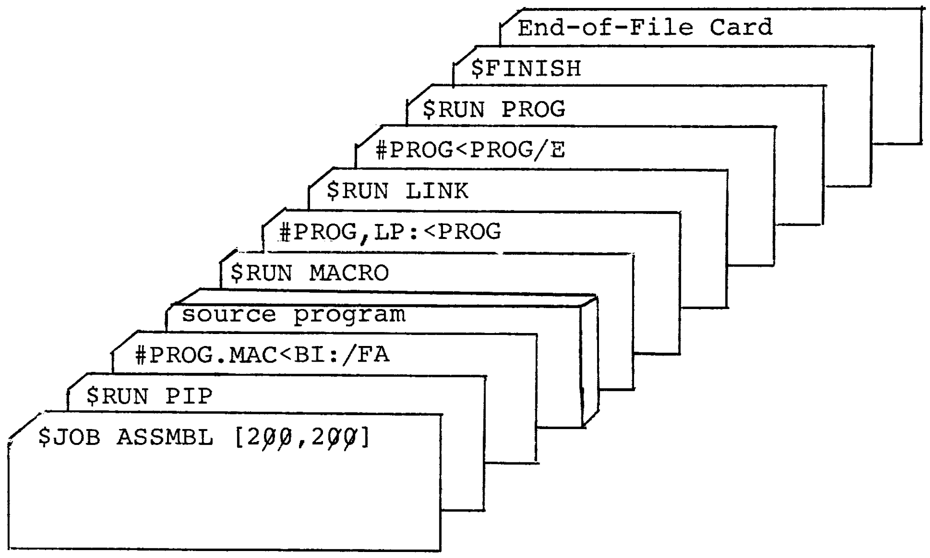


Figure 1-8 Sample MACRO-11 Job Deck

Note that the dataset specified as the output dataset (e.g., PROG.MAC) in the PIP command string must be specified as the input dataset in the MACRO-11 command string. /GO may be used in place of /E; the $RUN card would not be needed then.

1.4 BATCH CHARACTER SET

Batch uses the 64-character graphic subset of ASCII shown in Table 1-5. This table also indicates the correspondence between ASCII characters and DECØ29/Ø26 Hollerith codes, as they appear on punched cards.

Table 1-5

PDP-11 CARD CODES

CHARACTER	ASCII	DECØ29	DECØ26	CHARACTER	ASCII	DECØ29	DECØ26
SPACE	24Ø	NONE	NONE	@	3ØØ	8 4	8 4
!	Ø41	12 8 7	12 8 7	A	1Ø1	12 1	12 1
"	Ø42	8 7	Ø 8 5	B	1Ø2	12 2	12 2
#	243	8 3	Ø 8 6	C	3Ø3	12 3	12 3
$	Ø44	11 8 3	11 8 3	D	1Ø4	12 4	12 4
%	245	Ø 8 4	Ø 8 7	E	3Ø5	12 5	12 5
&	246	12	11 8 7	F	3Ø6	12 6	12 6
'	Ø47	8 5	8 6	G	1Ø7	12 7	12 7
(	Ø5Ø	12 8 5	Ø 8 4	H	11Ø	12 8	12 8
)	251	11 8 5	12 8 4	I	311	12 9	12 9
*	252	11 8 4	11 8 4	J	312	11 1	11 1
+	Ø53	12 8 6	12	K	113	11 2	11 2
,	254	Ø 8 3	Ø 8 3	L	314	11 3	11 3
-	Ø55	11	11	M	115	11 4	11 4
.	Ø56	12 8 3	12 8 3	N	116	11 5	11 5
/	257	Ø 1	Ø 1	O	317	11 6	11 6
Ø	Ø6Ø	Ø	Ø	P	12Ø	11 7	11 7
1	261	1	1	Q	321	11 8	11 8
2	262	2	2	R	322	11 9	11 9
3	Ø63	3	3	S	123	Ø 2	Ø 2
4	264	4	4	T	324	Ø 3	Ø 3
5	Ø65	5	5	U	125	Ø 4	Ø 4
6	Ø66	6	6	V	126	Ø 5	Ø 5
7	267	7	7	W	327	Ø 6	Ø 6
8	27Ø	8	8	X	33Ø	Ø 7	Ø 7
9	Ø71	9	9	Y	131	Ø 8	Ø 8
:	Ø72	8 2	11 8 2	Z	132	Ø 9	Ø 9
;	273	11 8 6	Ø 8 2	[	333	12 8 2	11 8 5
<	Ø74	12 8 4	12 8 6	\	134	Ø 8 2	8 7
=	275	8 6	8 3	]	335	11 8 2	12 8 5
>	276	Ø 8 6	11 8 6	↑ or ^	336	11 8 7	8 5
?	Ø77	Ø 8 7	12 8 2	← or _	137	Ø 8 5	8 2
{	173	12 Ø	12 Ø	}	175	11 Ø	11 Ø

NOTE

The code used is ASCII-68. Certain inconsistencies may appear, depending on the particular Ø29 keypunch used. On older keypunches, the ! and ↑ characters' codes are produced by punching the vertical bar (|) and ¬ keys, for codes 12 8 7 and 11 8 7 punches, respectively. On newer keypunches, the ↑ character is represented as a circumflex (^).

1.5 BATCH FEATURES

Batch incorporates most of the features of the Disk Operating System, providing well-tested, fully-documented facilities to the Batch user.

1.5.1 System Programs

All programs and utilities included in DOS are available to the Batch user. Note that the On-Line Debugging program (ODT) can be run through the keyboard only. Table 1-6 lists system programs usable in Batch, with the order number of the manuals in which they are documented. Chapter 4 describes the modifications made to these programs, to allow them to run in Batch.

TABLE 1-6

Batch System Programs

Program Name	Manual No.
FORTRAN IV	DEC-11-KFDA-D
Edit-11	DEC-11-ASDB-D
PIP, File Utility Package	DEC-11-PIDA-D
Link-11 Linker and Libr-11 Librarian	DEC-11-ZLDA-D
MACRO-11 Assembler and CREF Cross Reference Program	DEC-11-OMACA-A-D

1.5.2 Programmed Requests

The Disk Operating System Monitor Programmer's Handbook (DEC-11-MWDC-D) includes detailed descriptions of programmed requests, which are the means by which a user program calls for Monitor services. Some differences exist in the way certain programmed requests are processed in Batch mode, to allow for differences in the Batch and keyboard environments.

CHAPTER 2

BATCH COMMAND LANGUAGE

2.1 BATCH COMMAND SET

The information in this chapter is intended for users who require a more comprehensive understanding of the Batch operating system. It is assumed that the reader is familiar with the PDP-11 Disk Operating System, DOS, (as described in the publication, Disk Operating System Monitor Programmer's Handbook). The purpose of this chapter is to familiarize the user with the use of the command language, as it is applied in Batch operation. Commands which are invoked when the system is not in Batch mode, perform exactly as described in the DOS manual. The effect of a command in Batch mode depends on whether the command is received from the keyboard (following CTRL/C), or from the batch stream. Table 2-1 lists all commands, and their status in Batch mode.

TABLE 2-1

BATCH COMMANDS

Command	Action	
	From Keyboard	From Batch Stream
$ASSIGN	Assign a physical device and file name, to a dataset	Same as from Keyboard. See Note 1.
$BATCH	Invokes Batch monitor.	Invalid.
$BEGIN	Invalid.	Honored only when program loaded, and never started.
$CHANGE	Invalid.	Transfers batch stream to dataset specified.
$CONTINUE	Resumes program execution.	Ignored.
$DATE	Used as in DOS.	Can't contain a value. Causes date to be printed in the log.
$DUMP	Dumps core to line printer. Processing suspended until dump complete.	Same as from Keyboard.
$ECHO	As in DOS.	Invalid.
$END	As in DOS.	Invalid.
$EOD	Invalid.	Delimits physically contiguous, logically distinct data.
$FINISH	Invalid	Logical job delimiter.
$GET	Invalid.	Loads a program from the specified device.
$KILL	Terminates the current job.	Terminates the current program. See Note 2.
$MESSAGE	Invalid.	Outputs a message to the operator.
$MODIFY	As in DOS.	No display of location's previous contents.
$ODT	Invalid.	Invalid.
$OWN	Invalid.	Allows unformatted reads from batch stream.

TABLE 2-1

(cont'd)

Command	Action	
	From Keyboard	From Batch Stream
$PRINT	As in DOS.	Invalid.
$RESTART	Invalid.	Limited usefulness.
$RUN	Invalid.	Loads and starts a program.
$SAVE	Invalid.	Writes the program in core, onto the disk in loader format
$STOP	Invalid.	Invalid.
$TERMINATE	Terminates the batch session.	Invalid.
$TIME	Enter a value for TIME or display the time.	Request that TIME's value be output to the log. Entering time is illegal.
$WAIT	As in DOS.	Suspends job execution until CO is typed at keyboard.

Notes

1. $ASSIGN should not be used to assign devices to BATCH command datasets. Refer to Chapter 4 for details.

2. Other commands that force a "KILL", if read when the program is reading command string input, are; $RUN, $GET, $FINISH, $CHANGE, and $JOB.

2.1.1 Synchronous/Asynchronous Commands

Certain commands are treated as asynchronous commands, while the rest are dealt with synchronously. Asynchronous commands cause actio to be taken as soon as they are read, regardless of whether they are read from the Batch command input dataset (logical name CMI); from th system program command input dataset (PCI)[1]; or, from a user's datase

[1]Command datasets are discussed in Chapter 4.

Asynchronous commands include:

```
$BEGIN
$DATE
$DUMP
$EOD
$KILL
$MESSAGE
$MODIFY
$OWN
$RESTART
$SAVE
$TIME
$WAIT
```

Synchronous commands are those which are not executed immediately, unless they are read while the program is reading CMI. If read from PCI, or a user's dataset, they cause an "end-of-file" to be returned to the program; the command is held until a READ CMI is issued, at which point the command is executed. Synchronous commands include:

```
$ASSIGN
$CHANGE
$FINISH
$GET
$JOB
$RUN
```

All synchronous commands, except $ASSIGN, force a $KILL. $JOB also forces a $FINISH.

2.1.2 New Batch Commands

Five commands have been created for the use of Batch. They are:

1. $CHANGE
2. $EOD
3. $JOB
4. $ME
5. $OWN

These commands are only legal from the batch stream. They are fully described in later sections.

2.1.3 BATCH Command

The BATCH command invokes batch mode. It is described in Chapter 1.

2.1.4 TERMINATE Command

The TERMINATE (TE) command is entered through the keyboard, to end the batch session. It is legal only from the keyboard, following CTRL/C. If entered via the batch stream, the message INV CMD! is printed, and the current job is aborted.

2.2 JOB STATEMENT

Each job to be executed is headed by a job statement, in the following format:

$$\$\mathrm{JO[B]}\begin{Bmatrix}\Delta\\ ,\end{Bmatrix}\mathrm{name[uic]/sw1/sw2[log\ dataset]}$$

The dollar sign must be in position 1 (or card column 1, if a card reader is the input device). The first two letters of JOB are all that is required to identify this as the JOB statement, but it can be spelled out completely, for the sake of clarity. The $JOB field is delimited from following fields by a space or a comma.

2.2.1 Job Name

The job name is specified as a string of alphanumeric characters. Characters after the first six are ignored. A space, a slash, or a left bracket may terminate the job name.

Examples:

```
ΔnameΔ(whereΔ=space)
name/
name[
```

The job name is used by Batch for message processing and temporary file creation. If the user does not specify a job name, one is assigned by the system.

Default job names are assigned as sequential numbers, starting with ØØØØØ1. Each time the BATCH command is typed in, the sequence is reset. The name ØØØØØ1 is assigned to the first nameless job, ØØØØØ2 to the next nameless job, etc. Any number of user-named jobs can intervene between ØØØØØ1 and ØØØØØ2.

2.2.2 User Identification Code (UIC)

The user identification code is entered in the format shown below:

```
[nnn,nnn]
```

where "nnn" represents a string of one or more octal digits, up to a value of 376. *Note: A single zero cannot be used in place of* "nnn." Ø,nnn; nnn,Ø; *and* Ø,Ø *are all invalid. The values* Ø-1Ø *and* 377 *are reserved for the system.*

The value to the left of the comma represents the user-group number, while the value to the right represents the user's number within the group.

Example

```
[2Ø2,151]
```

This uic specifies user group 2Ø2, user number 151 within that group. If the user identification code is omitted, the default uic is used. The value of the default uic is installation specified; and observes the rule concerning the use of zero. If a uic is not specified in a $JOB statement, and the default uic is Ø, the job will not be executed.

2.2.3 Switches

Two switches may be used to specify actions. The first limits the length of time that the job can run, measured in clock-on-the-wall time, rather than in CPU time. The second switch allows the user to suppress the log. Each switch is preceded by a slash, followed by the keyword that identifies the switch.

The time-limit switch has the format:

```
/TI:hh:mm
```

where hh and mm are the values specifying hours and minutes, as decimal integers. A single value following /TI (e.g., /TI:15) indicates the number of minutes the job is to run. Thus, to indicate hours, but not minutes, the user must enter /TI:hh:ØØ, with a value for hh. The /TI switch in the BATCH command takes precedence over the time-limit specified in the $JOB command.

The "no-log" switch is specified simply as a slash, followed by the letters NL; i.e., /NL. In specifying this switch, the user indicates that he wishes no log to be produced for his job. The effect of /NL is to suppress the log dataset.

2.2.4 Log Dataset

An optional entry can be included specifying the dataset to be used for the log, for this job. The dataset specified overrides that given in the BATCH command.

2.3 COMMAND STATEMENTS

Command statements request that a specified action be performed. They may be directed to the Batch Monitor, to the Command String Interpreter, or to one of the system programs.

2.3.1 Monitor Command Statements

All Monitor commands are prefixed by a dollar sign in position 1. The following pages describe the commands directed to the Batch Monitor, not including concise commands (described in 1.2). Commands are arranged in alphabetic order. Only those commands that are valid or effective from the batch stream are discussed (refer to Table 2-1).

ASSIGN

2.3.1.1 $ASSIGN

Format

```
$AS[SIGN] dataset specifier, logical name
```

Purpose

This command assigns a physical device (and a file name, when the device is file-structured) to the dataset identified by "logical name". The format of "dataset specifier" is:

```
dev:filename.ext[uic]
```

where "dev" designates the device, and filename.ext[uic] designates the name, extension, and uic, if any, to be assigned to the file.

The "logical name" is the name that has been specified in the link block in the user's program.

NOTE: The $ASSIGN command should not be used with Batch command datasets (i.e., CMO, CMI, PCI, and CDI. These datasets are used for input to the Command String Interpreter (CMI), related output (CMO), input to system programs (PCI), and data input resulting from a program command (CDI). Refer to Chapter 4 for details).

The duration of an $ASSIGN depends on when it was issued. If issued at the job level, i.e., after $JOB, but prior to $RUN or $EXECUTE, an $ASSIGN remains in effect for the duration of the job, unless subsequently altered. If issued at the program level, an $ASSIGN is in effect for the duration of that program, unless changed during execution of the program.

Examples:

To assign a DECtape file named COM.BN to the dataset with the logical name ITR:

```
$AS DT:COM.BN,ITR
```

To assign a disk file to FORTRAN unit number 5:

```
$AS FILE.EXT,5
```

2.3.1.2 $BEGIN

Format

```
$BE[GIN][,address]
```

Purpose

The $BEGIN command starts execution of an already loaded program at the stated address. The address value, if specified, is an absolute octal value; if not stated, the normal start address is used.

The $BEGIN command is used only for programs that have been loaded (via $GET), but have not yet been started. The $BEGIN need not immediately follow the $GET. The effect of the $GET... $BEGIN sequence is the same as the $RUN command; the main purpose is to allow the user to insert changes into the program, which has been loaded, but not begun. The $MODIFY command is used to make these changes (see 2.3.1.11).

Example

```
$BEGIN
```

Start executing a program at the normal start address.

2.3.1.3 $CHANGE

Format

    $CH[ANGE] dataset

Purpose

Changes the batch stream input to the dataset specified. This permits data, source programs, etc., to be stored on datasets other than the one used normally for batch input, and then to be read in during execution of a job.

$CHANGE command is a synchronous command. When the $CHANGE command is honored, the batch stream is read from the secondary dataset. When end-of-file is sensed on the secondary dataset, command input is resumed from the primary dataset, at the point following the $CH command.

Example

```
          DATASET A                               DATASET B
          (Primary)                              (Secondary)
    1.  $JOB AAA[2ØØ,2ØØ]                      $RUN FORTRN
    2.  $RUN MACRO                             #PROGA,LP:<PROGA
    3.  #FILEA<FILEA                                  .
    4.  $CHANGE DATASETB                              .
    5.  $RUN LINK                                   (EOF)
    6.  #FILEA,LP:<FILEA,PROGA,FTNLIB/L/E
    7.  $FINISH
```

In this example, when EOF is encountered on dataset B, command input is resumed at command 5 of dataset A ($RUN LINK).

NOTE

$JOB, and $CHANGE are not legal in the secondary dataset. The job will be aborted if either is encountered.

2.3.1.4 $DATE

Format

```
$DA[TE]
```

Purpose

Requests that the current date be included in the job log.

The date will be printed in the dd-mmm-yy format. When entered via the batch stream, the $DATE command may be used solely to place the date of the job's execution in the log.

When the $JOB card is processed, the date and time are put in the log.

The user can enter a date value through the keyboard, while Batch is running. To do this, type

```
CTRL/C
.DATE dd-mm-yy)
```

putting the correct date value in place of

dd-mmm-yy.

DUMP

2.3.1.5 $DUMP

Format

$$\text{\$DU[MP],LP:}\left[\left[\text{,O}\right]\ \left[\text{, }\left\{\begin{array}{c}\text{start addr}\\ \underline{\emptyset}\end{array}\right\}\ \left[\text{,end addr}\right]\right]\right]$$

Purpose

The $DUMP command causes an absolute copy of a specified core area to be written out of core to the line printer. If no arguments, other than devicename, are supplied, values are assumed by default; i.e., "O", dump from core to the line printer; starting at address Ø. If no end address is specified, the highest word in memory is the default value. A $DUMP command is valid at any time; if issued during program execution, operations are suspended for the time needed to complete the dump. $DUMP can be entered through the keyboard while in Batch mode.

2.3.1.6 $EOD

Format

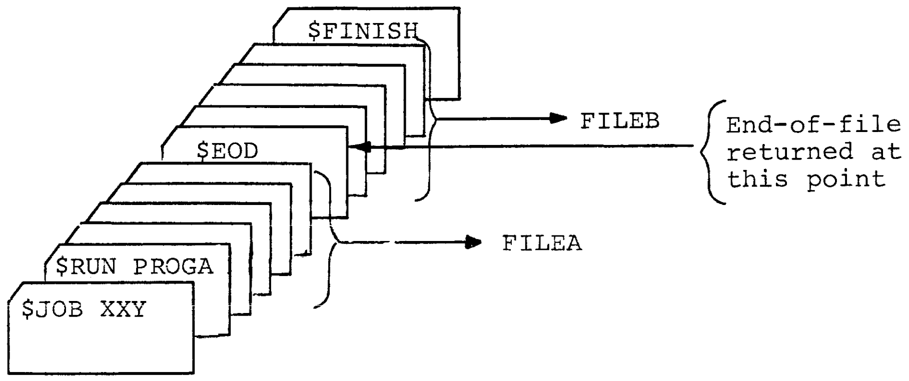
```
$EO[D]
```

Purpose

The $EOD command stands for "End-of-Data". It is the command that delimits groups of data statements that are logically distinct, but physically contiguous.

When this statement is encountered in the batch stream, an "end-of-file" is generated and returned to the current program. This "end-of-file" indicates that the data that follows is logically distinct from data the program has read to that point.

$EOD allows data to be stacked in a deck in physically contiguous fashion, while the program treats each group of cards as a logical unit, as the "end-of-file" is sensed. In the example below, the program PROGA processes data contained in two logically distinct datasets: FILEA, and FILEB. The $EOD card signals the end of FILEA, by returning "end-of-file" to PROGA when the last record in FILEA has been read.



NOTE: $EOD provides a logical end-of-file; it is not the same as the physical end-of-file card which is required to signal the end of a card file. (Refer to Appendix B.)

2.3.1.7 $FINISH

Format

```
$FI[NISH]
```

Purpose

The $FINISH command is used to delimit a job. When a $FINISH is detected, it signifies that the current job is ended. There are no arguments associated with $FINISH.

Processing continues with the next $JOB statement. *Note that the* $FINISH *command cannot be entered through the keyboard while the system is in batch mode.*

To terminate the batch stream from the keyboard, type:

```
CTRL/C
.TE )
```

2.3.1.8 $GET

Format

$$\text{\$GE[T] program specifier} \left[\text{/DU} \left[\begin{Bmatrix} \text{:}\underline{\text{PR}}\text{[OGRAM]} \\ \text{:}\overline{\text{AL}}\text{[L]} \\ \text{:}V_1\text{:}V_2 \end{Bmatrix} \right] \right]$$

Purpose

This command causes the program to be loaded into core from a specified device. The program specifier entry can include the device identifier, and the filename (and extension, if any) and uic of the program to be loaded. A $BEGIN command would, at some point, normally follow the $GET, to start the program. (Refer to 2.3.1.2.) $GET can also be used in conjunction with the $SAVE command (see 2.3.1.15).

A "dump-on-error" switch (/DU) is available, allowing the user to obtain a dump of a specified area, in case an error occurs in his program. One of three values may be specified with the dump switch:

PR[OGRAM]	- dump the program area
AL[L]	- dump core in its entirety
V_1:V_2	- dump the area delimited by the octal addresses specified for V_1 and V_2. V_2 must be greater than V_1 and must be even (i.e., word boundaries).

The default value is PR. The dataset to which the dump is made is system-defined.

The "program area" depends on the location of the stack pointer.

1. If the stack pointer is below the load address, the program area is from the stack pointer to the top of core;
2. If the stack pointer is not below the load address, the program area is from the load address to the top of core.

/DU forces an automatic dump in the event of an error, thus providing the Batch user a means of debugging his program. Refer to Chapter 3 for the format of error dumps.

2.3.1.9 $KILL

Format

```
$KI[LL]
```

Purpose

The $KILL command terminates the current program, stops I/O and closes all open files. Processing continues at the next command prefixed by a $. No arguments are specified for $KILL.

The $KILL command can be entered via the keyboard to abort the job while in batch mode: for example, to abort a job that is threatening to pre-empt system resources to the detriment of other jobs. Type in the sequence

```
CTRL/C
.KILL ⤶
```

The following commands force a KILL:

```
$CHANGE
$FINISH
$GET
$JOB
$RUN
```

2.3.1.10 $MESSAGE

Format

$ME[SSAGE]

Purpose

The $ME statement is used to send a message to the operator. A single message consists of one line. To send a message from the batch stream, the user inserts commands in the following format:

$ME $\left\{\begin{matrix}\Delta \\ ,\end{matrix}\right\}$ any text

The text portion of the lines can be any message the user wishes passed to the operator. The message is printed on the teleprinter.

Example of the $ME Usage:

Message lines in batch stream (job name TBLT);

```
$ME MOUNT DECTAPE LABELLED XYZ ON UNIT 1
$ME DO NOT WRITE ENABLE
$WAIT
```

Keyboard output;

```
TBLT  : MOUNT DECTAPE LABELLED XYZ ON UNIT 1
TBLT  : DO NOT WRITE ENABLE
AØ5Ø ØØØØØØ
$
```

The keyboard output includes the job name (TBLT) in the message line. The Action message (AØ5Ø ØØØØØØ) indicates that a $WAIT is in effect; the program is suspended until the operator types CO on the same line as the $.

The $WAIT statement is optional; if omitted, the message is printed at the keyboard, but the program continues without suspension. (Of course, the AØ5Ø ØØØØØØ, $ lines are not printed, since no operator response is expected.)

2.3.1.11 $MODIFY

Format

$MO[DIFY] $\left\{\begin{matrix}\Delta\\,\end{matrix}\right\}$octal addr:new contents

Purpose

This command provides a way of changing the contents of a memory location. Whatever was in the location is altered to the value specified in the "new contents" field. The system makes no provision for displaying the previous contents of the address. The value specified in the "octal addr" field must be an even number (i.e., aligned on a word boundary), and must not exceed 16 bits. Locations in the resident Monitor may not be MODIFied.

Example

```
$MO 2164Ø:16Ø4Ø
```

This changes the contents of location 2164Ø to 16Ø4Ø.

NOTE

The location specified must not fall within the area occupied by the Monitor or the job will be aborted.

2.3.1.12 $OWN

Format

```
$OW[N]
```

Purpose

This command causes the system to enter a mode of operation called "own mode", which allows data to be read in unformatted mode. Statements in the batch stream are treated as pure data; thus, special characters, such as $, #, and *, which might ordinarily cause some Monitor action to occur, are treated as datε rather than control characters.

Currently, the $OWN command can be used only when the batch stream device is the card reader.

Return from own mode to normal input mode is effected by placing an end-of-file card at the end of the data.

NOTE

The characters $, #, and * can also be read as data, rather than control characters, by placing an apostrophe in the first position of the line (refer to section 2.4).

2.3.1.13 $RESTART

Format

```
$RE[START][,address]
```

Purpose

This command permits a program to be restarted. As shown, the user may optionally supply an address at which the program is to be restarted. Normally, a restart address will have been specified by the program. It is recommended that the address option for $RESTART be used with care.

$RESTART is valid only when the program is already loaded. Before the program is restarted, the stack is cleared, any current I/O is stopped, and all internal busy states are removed. Buffers and device drivers set up for I/O operations will, however, remain linked to the program for future use.

The $RESTART command is invalid if a restart address has not been specified, either by the program, or by an address field with the command itself. $RESTART may not be entered from the keyboard while the system is in batch mode.

RUN

2.3.1.14 $RUN

Format

$$\text{\$RU[N],program specifier}\ \left[\text{/DU}\left\{\begin{array}{l}:\underline{\text{PR}}\text{[OGRAM]}\\ :\text{AL[L]}\\ :V_1:V_2\end{array}\right\}\right]$$

Purpose

The $RUN command causes a named program to be loaded from a specified device, and started at the normal address.

The program specifier provides the name of the program, and the device from which it is to be loaded, and, optionally, a user identification code that is associated with the program.

As with the $GET command, a "dump-on-error" switch may be included, in the format

$$\text{/DU}\left\{\begin{array}{l}:\text{PR[OGRAM]}\\ :\text{AL[L]}\\ :V_1:V_2\end{array}\right\}$$

where:

PR[OGRAM]	means "dump the program area" (see the description for $GET, section 2.3.1.8);
AL[L]	means "dump all of core"; and
$V_1:V_2$	means "dump the area bounded by the octal addresses specified for V_1 and V_2. V_1 and V_2 must be even values (i.e., word boundaries), and V_2 must be greater then V_1.

The default area is PROGRAM. If no /DU switch is included, no dump-on-error occurs. The dataset to which the dump is made is system-defined.

Example

```
$RU DT:PGM/DU
```

The program named PGM is loaded from DECtape, and started. The dump on error specification requests that the program area be dumped (by default).

2.3.1.15 $SAVE

Format

```
$SA[VE][,dataset specifier [/RA:low:high]]
```

Purpose

The $SAVE command allows a program to be saved in loader format. It is used after a program has been loaded into core, prior to starting the program. The program is copied onto the device specified in the dataset specifier, under the name that is included in dataset specifier, if any.

The $SAVE command may be used only if the program was never started. A common use of $SAVE is to $GET a program (load it), insert fixes through $MODIFY commands, and then place the altered program onto secondary storage through a $SAVE command.

If no dataset specifier is included, the current program will be saved on the system disk, under the name SAVE.LDA. Any file previously saved under this name will first be deleted.

The /RA switch is included so the user can save an area other than that comprised by his current program. If he wishes to save only the current program area, this switch is omitted. Including the /RA switch allows the specified area to be saved. The addresses specified for /RA must be valid octal word boundary addresses.

The command will be rejected if an additional 256 word buffer cannot be allocated from free core.

Example

```
$SA,REG.LDA
```

The $SAVE command in this example causes the current program to be saved on the system disk, under the name REG.LDA.

2.3.1.16 $TIME

Format

```
$TI[ME]
```

Purpose

Including the $TIME command provides a means of obtaining the time-of-day in the output job log. It does not permit the user to specify a time from the batch stream. Attempting to do so is illegal. (The current job will be aborted.)

To enter a time value, the user should type, at the keyboard,

```
CTRL/C
$TIME hh.mm.ss)
```

This is a valid entry from the keyboard, while the system is operating in batch mode.

The time-of-day is also placed in the job log by the $FINISH and $JOB commands. The $JOB command also includes the current date.

2.3.1.17 <u>$WAIT</u>

<u>Format</u>

```
$WA[IT]
```

<u>Purpose</u>

This command suspends processing, and causes the Action message

```
A050 000000
$
```

to be printed at the keyboard. It is usually used in conjunction with the $ME command. To resume operation, type CO.

2.3.2 Input to Command String Interpreter

The Command String Interpreter (CSI) accepts as input, command strings consisting of dataset specifications. The purpose of a command string is to establish the datasets to be used by a particular program, for input and output. In Figure 1-4, the third card (#MATX,LP:<BI:) is a command string. As indicated there, the first character must be #, in column 1 (or in line position 1, if input is not from a card). The format of a CSI command is:

```
          #output dataset(s)<input dataset(s)
```

where a dataset can be specified as

dev:filename.ext [uic]/sw_1:v_1...v_n/sw_n:v_1...v_n

Each dataset specification is delimited by a comma. The elements comprising a dataset specification provide information concerning the dataset's location, its file name and extension (if it's a file), the user identification code associated with the file, and any switches that may be used to specify particular actions to be performed.

Device specifiers are selected from those listed in Table 1-2. (Refer to Chapter 1.) The ability to include the pseudo-device specifiers (BI and SY) is a feature of Batch that provides device-independence when specifying datasets to the Command String Interpreter. It allows a dataset to be specified without requiring that the user know what device may actually be used at job execution time.

For example, a user may have his source data on cards, but, because of the greater speed to be gained by reading the data from a faster device, he may transcribe the data onto another storage medium, such as disk. He would then specify the disk to be the batch stream device.

```
          $RUN FORTRN              - run the FORTRAN IV Compiler;
          #DK:PRG,LP:<BI:          - specify that the input dataset
                                     is located in the batch stream.
```

The same command string can be used, whether the batch stream is coming from disk, magnetic tape, DECtape, paper tape, or cards. By specifying BI, the user has ensured that the command string is valid for all these devices; there is no need to change the card to match the specific device.

The SY device specifier is used to designate the system-residence device, as in the following example:

```
#SY:FILE.FTN<PR:
```

In this example, a command string to PIP specifies that a dataset is to be input from the paper-tape reader, and output to the system-residence device. This command string is valid, whatever the system-residence device may be when PIP is executed.

2.3.3 System Program Commands

Commands that are directed to system programs (refer to Table 1-6, Chapter 1) are identified by an asterisk in position 1. For example, to issue the "Insert" command to Edit-11, the user must include a command in the format

```
*I
```

followed by the text to be inserted. Refer to the appropriate manual for details on the commands used in system programs.

2.4 READING CONTROL CHARACTERS AS DATA

The characters $, #, and *, appearing in the first position of a line (or card column 1), are interpreted as control characters, and are stripped off before the remainder of the line is passed to its destination (Monitor, Command String Interpreter, or system program). It may happen that the user wishes to include one of these characters as actual data, to be passed along with the rest of the data on the line, rather than having it stripped off. To do this, place an apostrophe in the first position; e.g.,

```
'$AMT 1ØØ
```

which causes the line to be passed as $AMT 1ØØ (the apostrophe is stripped).

If the apostrophe is not found in column 1, but the $ is there instead, the card would be treated as a command to the Monitor.

Valid control statements <u>can</u> be included, to cause the Monitor, CSI, or system program to take a desired action. Thus, a deck of cards being read by a user program may include a statement such as

```
$RUN PIP
```

to invoke the Peripheral Interchange Program for whatever reason the user may have. When the $RUN PIP card is encountered, an EOF is returned, and the card is held until a READ is issued to the Command Input dataset (CMI); at this point, the $RUN PIP card is passed to the Monitor, which causes the user program to be terminated, and PIP to be loaded and executed. (Refer to the discussion of synchronous/asynchronous commands, in Section 2.1.1.)

CHAPTER 3

INPUT/OUTPUT

3.1 BATCH INPUT

Input to Batch can be from any PDP-11 device that can perform the input function. Data can be read from the batch stream in one of two ways: "normal" mode, or "own" mode.

3.1.1 Normal Input Mode

In most cases the normal input mode is employed when reading data from the batch stream. Data read in normal mode must be formatted data. Attempting to read unformatted data, if not in "own" mode, results in a fatal error, aborting the job. (See the DOS Monitor Programmer's Handbook for a definition of formatted data.)

This requirement stems from the need to check the first position on each line for the presence of a control character. Because it is impossible to determine the beginning and end of a line in unformatted data, a situation could arise where the control card could be inadvertently bypassed, causing unpredictable results. *Note: All formatted READS from the batch stream must have a byte count of at least 83 specified in the line buffer header maximum byte count word.*

3.1.2 Own Mode

It is occasionally necessary to read in unformatted mode from the batch stream; e.g., when translating EBCDIC characters to ASCII. This is permitted by means of the $OWN control card, which indicates to the Monitor that all characters read from that point, until the next physical end-of-file (EOF) card (that terminates "own" mode), may be read as unformatted data. The characters $, #, and * are not treated as control characters, but as data.

The physical end-of-file (EOF) card statement must be included at the end of the user's "own" data, to avoid the possibility of failing to recognize a control card. The operator must place an EOF card at the end of each job, to prevent the next job from being read as data, if the prior user forgot to terminate "own" mode. For added safety, the user should place an EOF card immediately ahead of his $JOB card.

3.2 BATCH OUTPUT

Output from Batch includes program listings, output associated with system programs (such as load maps), the job log, and dumps.

3.2.1 Job Log

The job log is the record of events that occurred during execution of the job: the control cards processed, commands read, and error messages generated. The first line of the job log contains the image of the $JOB command, as specified by the user, the date, and the time. This is followed by a sequence of images of control cards that were read in and processed up to the point at which the $FINISH command was read, or a fatal error occurred. Any error, warning, or informative messages are included in the log as they are encountered. A log is produced for all jobs, unless the log-suppress switch (/NL) was specified in the $JOB card, provided that a log dataset was included in the BATCH command , or in the $JOB command.

3.2.2 Dumps

If the user has specified the /DU switch on the $RUN or $GET card for the program, and an error occurs, a dump of the area specified in the /DU switch is produced. The first page of a dump, the Header page, consists of a summary of information regarding the dump itself. It is formatted as shown in Figure 3-1.

```
          * * * * * * DUMP OF HEADER FOLLOWS * * * * * *

 STARTING WORD OF DUMP = nnnnnn
 NO. OF BYTES DUMPED   = nnnnnn
 RØ = nnnnnn
 R1 = nnnnnn
 R2 = nnnnnn
 R3 = nnnnnn
 R4 = nnnnnn
 R5 = nnnnnn
 SP = nnnnnn
 PC = nnnnnn
 PS = nnnnnn
 HIGH ADDRESS = nnnnnn
 LOW ADDRESS = nnnnnn

 DUMP IDENTIFIER MESSAGE
```

Figure 3-1 Sample Header Page

The values of "nnnnnn" are given in octal, and are left-justified (e.g., the value 477, in the NO. OF BYTES DUMPED entry would appear as:

```
NO. OF BYTES DUMPED = 477).
```

The subsequent pages of the dump comprise the area specified in the /DU switch. As shown in Figure 3-2, each group of four lines describes $1ØØ_8$ locations, and is headed by the flag --- n---, where n corresponds to the first location in the group; Ø, 1ØØ, 2ØØ, etc.

Each line in the dump contains the contents of eight words, represented as octal values. If a sequence of lines contain all zeroes, the first line in the sequence is printed in its entirety.

The next line is printed with asterisks in the first position, e.g.:

```
4Ø: ******
```

Subsequent lines in the sequence are omitted altogether. The next non-zero line is printed.

At the end of each line is a field of 16 ASCII characters, which is the ASCII contents of each byte. ASCII equivalents of characters that are not within the printable range are mapped into the printable range, converted to upper case (if necessary), and printed.

Figure 3-2 illustrates the form and content of a dump.

```
                    * * * * * * DUMP OF DATA FOLLOWS * * * * * *



---156700---
060: 000000 000000 000000 000000 000000 000000 000000 000207: @@@@@@@@@@@@@@G@


---157000---
000: 025266 000002 000764 140003 000000 000000 040515 051103: VJB@TAC@@@@@MACR
020: 000117 000000 000034 050553 071330 026105 001504 050114: O@@@C@KCXRELDCLP
040: 036072 026103 026507 000012 047506 052122 047122 000000: Z<CLGMJ@FORTRN@@
060: 000036 023752 077736 026105 001504 050114 036072 001510: ^@JG^CELDCLPZ<HC


---157100---
000: 044502 006472 000012 050103 000131 000000 000000 000024: BIZMJ@CPY@@@@@T@
020: 062570 000000 036103 006507 000012 042504 000114 000001: XF@@COGMJ@DFL@A@
040: 000000 000026 062570 000000 027503 042504 043454 005015: @@V@XF@@CODFLGMJ
060: 044504 000122 000000 000000 000040 062570 000000 001504: DIR@@@@@@@XF@@DC


---157200---
000: 041113 036072 001504 054523 027472 044504 043454 005015: KBZ<DCSYZ<DILGMJ
020: 054105 000000 000000 001430 000000 001430 177777 000000: EX@@@@XC@@XCCC@@
040: 001430 177777 157306 000000 025403 005266 157316 157316: XCCCFA@@CKVJNANA
060: 015572 001402 104001 177776 001066 000011 000000 000004: ZIBCAHACVRI@@@D@


---157300---
000: 157342 156776 001000 015676 007150 015574 000001 007150: BAAI@BAIHNCIA@HN
020: 000400 000000 000000 000000 000001 000001 157422 000004: @A@@@@@@A@A@RCD@
040: 000000 177777 156776 005000 005001 005002 005003 005004: @@CCAI@IAJBJCJDJ
060: 012705 000001 005046 005046 005046 104064 010504 013700: ELA@FJFJFJTHDC@W


---157400---
000: 000040 016046 000004 162716 001000 012746 177777 005046: @@FOD@NE@BFLCCFJ
020: 104064 010503 012746 000400 012746 000500 005046 104064: THCCFL@AFL@AFJTH
040: 010502 010546 010546 005046 104064 010501 010015 000000: BCFCFCFJTHACMF@@
060: 000000 000000 000102 000003 000000 000000 157700 000000: @@@@B@C@@@@@@C@@


---157500---
000: 010706 024646 010705 062705 000112 005001 013716 177570: FCFIECEFJ@AJNWXC
020: 006016 103402 005016 000403 006316 001001 010116 005000: NLBGNJCANLABNP@J
040: 004715 105303 001374 004715 004767 000074 010402 162702: MICJCBMIWIC@BCBE
060: 000004 022702 000002 001441 004767 000054 061604 010401: D@BEB@ACWIL@DCAG


---157600---
000: 004715 002004 105700 001753 000000 000751 110321 000770: MIDC@KKC@@IAQPXA
020: 016703 000152 105213 105713 100376 116303 000002 060300: CIJ@KJKKA@CCB@@@
040: 042703 177400 005302 000207 012667 000046 004715 010304: CF@CBJG@WLF@MIDP
060: 004715 000303 050304 016707 000030 004767 177752 004715: MIC@DPGIX@WIJCMI
```

Figure 3-2 Sample Data Dump

CHAPTER 4

BATCH PROGRAMMING CONVENTIONS

This chapter is aimed at programmers responsible for writing, modifying, or maintaining programs that function in the Batch operating system environment. Familiarity with DOS is assumed.

To function properly in Batch, programs must observe certain conventions. Currently existing programs should be modified to conform to these conventions, if the user intends them to operate in the Batch environment.

As an aid in making the modifications, assemble the programs to be modified, with MACRO-11, and obtain a CREF listing. The cross-reference data will help in locating and examining relevant link blocks and file blocks.

4.1 COMMAND DATASETS

Command String Interpreter input, and related output must be accomplished by datasets with the logical names: CMI (for command input); and CMO (for command-related output).

Command input to system programs, such as the *I command to Edit-11, must be through a dataset with the name PCI. Data read as the result of processing a command such as *I, must be read from a dataset named CDI. That is, the I insert command (*I) is read from PCI, while the text to be inserted is read from CDI.

4.1.1 Command String Input (CMI)

The dataset named CMI is used for all Command String Interpreter input; i.e., all commands with a # symbol in line position (or card column) 1. When reading CMI, a .WAIT should follow the .READ, and a test for end-of-data (EOD) should be made; if EOD has occurred, the Monitor EXIT EMT should be issued.

4.1.2 Command Output (CMO)

The dataset with logical name CMO is used for:

a. All Command String Interpreter related output (such as syntax error announcements, and the # symbol);

b. all responses to program command input (e.g., *);

c. all error logging.

A default must be specified for the CMO physical device. This default must be

```
KB:
```

so that the program will run in either DOS or Batch mode.

NOTE: Because the physical devices for the datasets used for command string input and output will, in all likelihood, not be the same; it is recommended that a .WRITE to CMO be followed by a .WAIT, before issuing a .READ to CMI, and vice-versa.

4.1.3 Program Command Input

All program command input must be entered via a dataset named PCI. That is, all commands prefixed by * must be read from PCI. When end-of-data (EOD) is detected from PCI, the proper procedure is to clean up the current Command String Interpreter request (# command), and read the next # command. Input to programs, other than program commands, such as an insert to Edit-11, must be entered via a link block that has a logical dataset name CDI. When EOD on CDI is detected, the current *command processing has finished. The next * command should be read from PCI.

4.2 READS FROM BATCH STREAM

As described in Chapter 3, all READS from the batch stream must be formatted, unless the $OWN command has been issued. All formatted READS from the batch stream must have a byte count in the line buffer header of $\geq 83_{1\emptyset}$. This requirement precludes corruption of commands that may be read by the user. (If the byte count is less than $83_{1\emptyset}$, a command on a card might be truncated before being read in its entirety.)

4.3 PSEUDO DEVICE SPECIFIERS

The device specifiers BI and SY are used when the user wishes to call for the Batch Input device, or the SYstem residence device, but he does not know which device is actually being used. Since these specifiers do not call for a "real" device, in the way that CR or DF do, they are termed "pseudo" device specifiers. They allow the system to supply the actual device that is being used at job execution time, in place of BI or SY. Thus, the same control cards can be used, regardless of the particular device being employed.

4.4 USE OF $ASSIGN

The ASSIGN statement must observe the rules listed below:

1. $ASSIGN must not be used with any Batch system program's logical dataset name. I.e., the user must not assign a dataset to CMI, CMO, CDI, or PCI.
2. An $ASSIGN that is made at the job level is global to the job, thus:

   ```
   $JOB MAC [2ØØ,2ØØ]

   $ASSIGN DTØ:CRT,RDO
   ```

   causes a file named CRT, which is on DECtape, to be assigned to the dataset with logical name RDO. This assignment, if not altered by a later ASSIGN in the job, remains in effect for the duration of the job (MAC).
3. An $ASSIGN that is made at the program level remains in effect (if not subsequently altered) for the duration of the program.

   ```
   $GET PROG

   $ASSIGN DK:MTX.OBJ,DCL

   $BEGIN
   ```

   The assignment of file MTX.OBJ, on the RK11 disk, to dataset DCL is in effect for the duration of PROG.

4.5 NOTE PERTAINING TO .CSI2 RETURN CONDITIONS

The user should note that on return from .CSI2, the top of the stack may have bit 2 set. Bit 2 is set when a default device is returned by the Command String Interpreter; i.e., the user has not specified a device in a command string, but has chosen to use the default device, instead.

As documented in the DOS Monitor Programmer's Handbook, the user is only required to check bits Ø and 1. In cases where this is done by checking bit 1 (to determine that no error occurred), and then checking the value of the word for a zero or non-zero value, the presence of a "1" in bit 2 may lead to erroneous assumptions.

4.6 ERROR HANDLING

The Command Output dataset (CMO) must be used for output of all error announcements that come directly from a system program, rather than via an IOT. Direct error announcements include announcements of command string syntax errors, and supplementary information (such as file names) concerning error announcements made through an IOT.

An EMT has been incorporated into the Batch system, to allow the currently running program (system or user), to request that lines in the batch stream be bypassed, until a specified type of control card is encountered. For example, if a command string syntax error occurs, it may be desired to bypass all following statements up to the next $ or # statement. This EMT is incorporated by including the following:

```
MOV #CODE,-(SP)
EMT 67
```

where CODE's value determines the next statement type to be read (not bypassed):

CODE	STATEMENT TYPE
Ø	$
1	$ or #
2	$, #, or *

Any other value for CODE is invalid, and causes a fatal error (FØ53).

The DOS convention of announcing command syntax errors, by printing the command as far as the point of the error, followed by ? is still used in Batch, but the EMT shown above must be included to cause the batch stream to be bypassed from the point of the error until a line starting with $, or # is found. (CODE = 1) EMT 67 is a NOP when the system is not in Batch mode.

APPENDIX A

OPERATING THE CARD READER

A.1 CR11 CARD READER

The usual medium of input to Batch will be the CR11 card reader. Two models are available for use with Batch: CR-11A, and CR11-B. Operating procedures for each are detailed below.

A.1.1 CR11-A Card Reader

Figure A-1 illustrates this model. To operate CR11-A:

1. Turn power switch ON.
2. Remove card-deck weight from input hopper.
3. Place card deck in input hopper. Cards should be face down, with the notched corner to your left. Edges of all cards should be aligned evenly.
4. Replace card-deck weight on top of deck.
5. Press MOTOR START button.
6. Press READ START button.

The unit is ready for operation if the green lights associated with MOTOR START and READ START are lit; no red lights should be on. If a red light is on, do not attempt to enter the BATCH command at the keyboard, because the card reader is not ready.

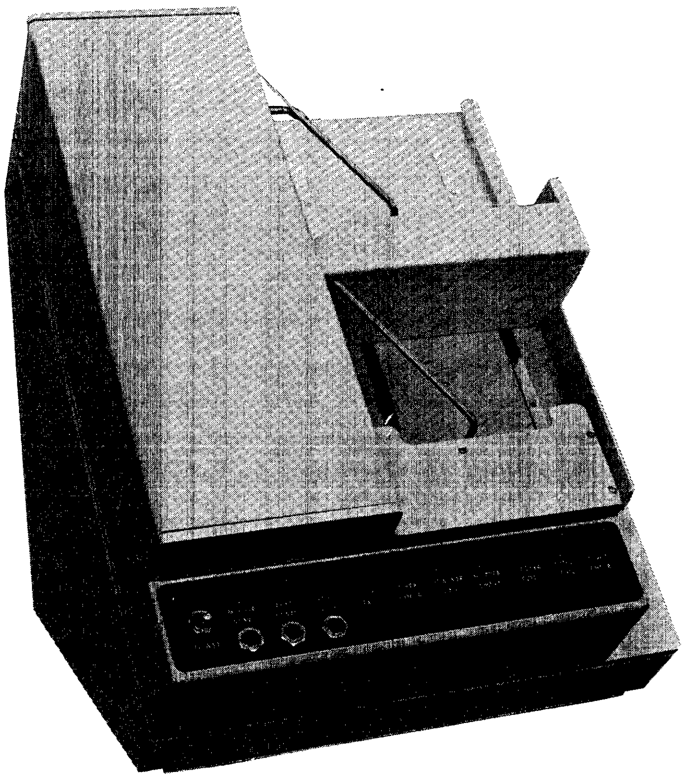


Figure A-1 Model CR11-A Card Reader

A.1.2 CR11-B Card Reader

Model CR11-B is shown in Figure A-2. It is operated as follows:

1. Turn power on (switch located in upper left corner in back of the unit).
2. Remove card-deck weight from input hopper.
3. Place deck in hopper, with cards evenly aligned, face down, notched corner to your left.
4. Replace card-deck weight on top of deck.
5. Press RESET button.

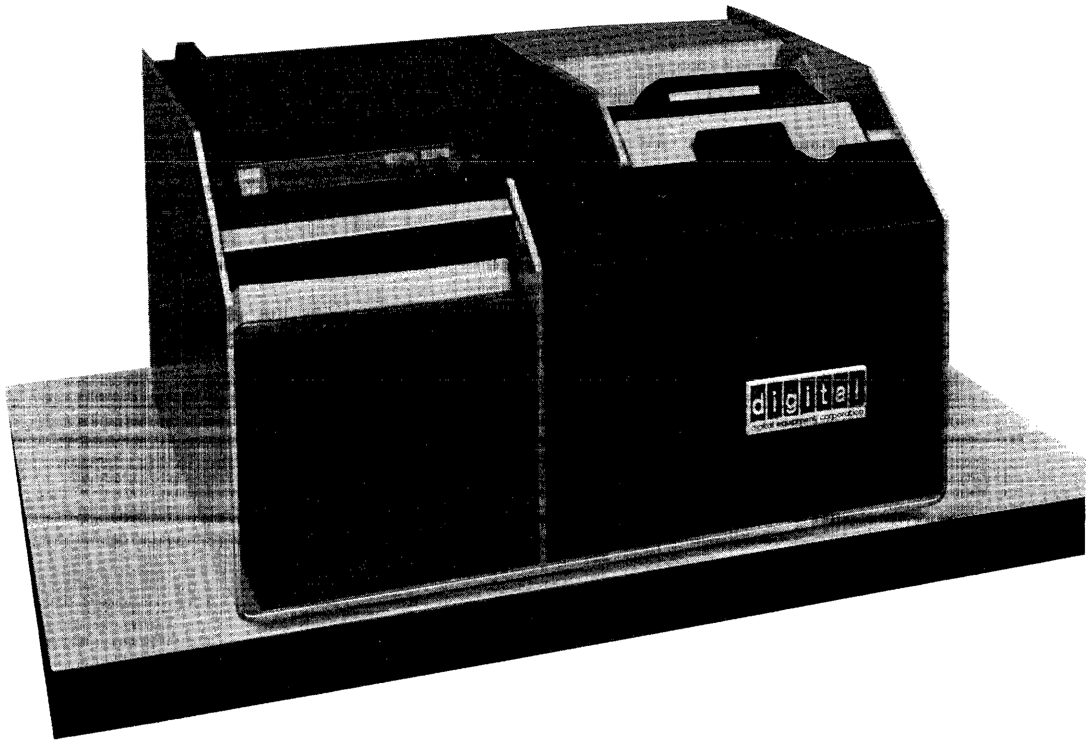


Figure A-2 Model CR-11B Card Reader

APPENDIX B

MULTIPUNCH CARDS

B.1 BRACKETS []

B.1.1 Ø29 Code

Figure B-1 illustrates the left bracket (12-8-2) in column 5, and the right bracket (11-8-2) in column 10. Note: 12=&, 11=-.

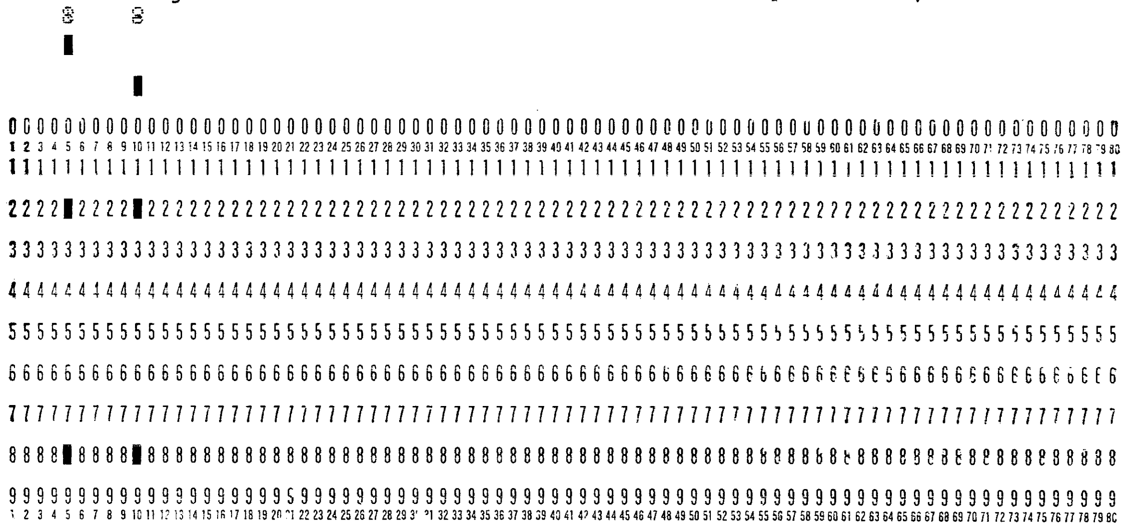


Figure B-1 Ø29 Bracket Multipunches

B.1.2 Ø26 Code

Figure B-2 illustrates the left bracket (11-8-5) in column 5, and the right bracket (12-8-5) in column 10.

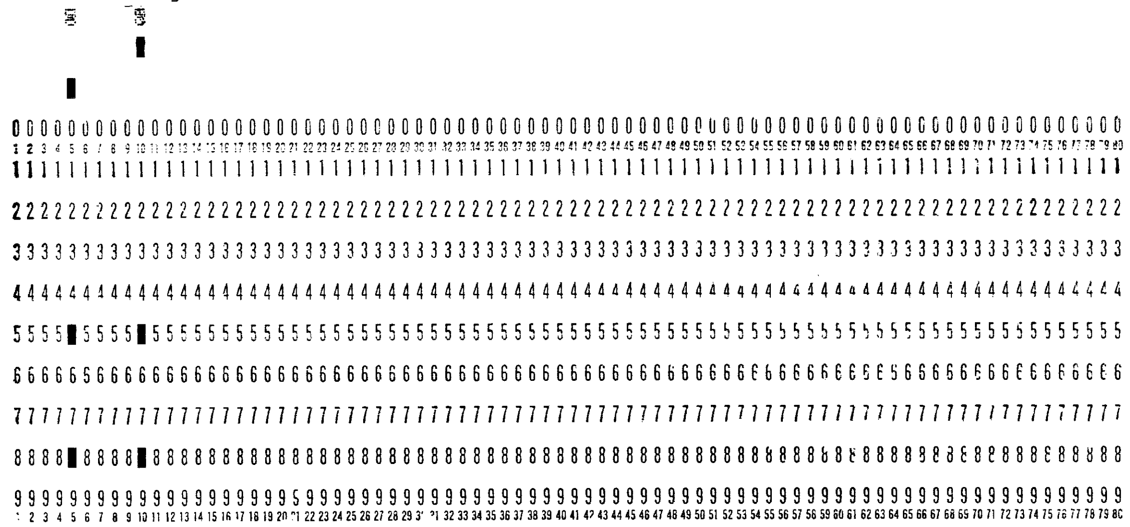


Figure B-2 Ø26 Bracket Multipunches

B.2 END OF FILE CARD

The End-Of-File card is multipunched as 12-11-Ø-1-6-7-8-9 in column 1. See Figure B-3.

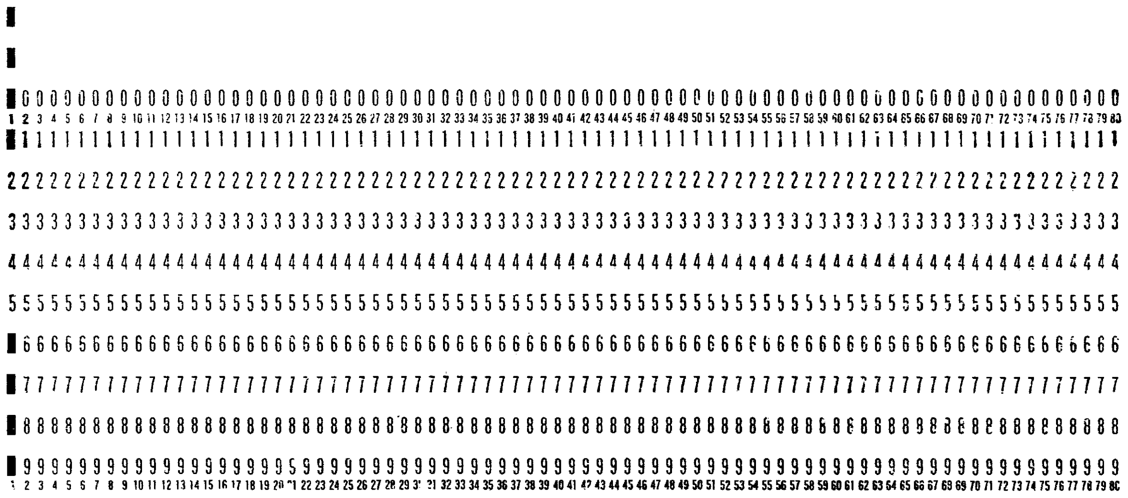


Figure B-3 End-of-File Card

B.3 BLANK SUPPRESS

The Blank Suppress card has a multipunch of 12-11-Ø-7-8-9 in column 1. See Figure B-4.

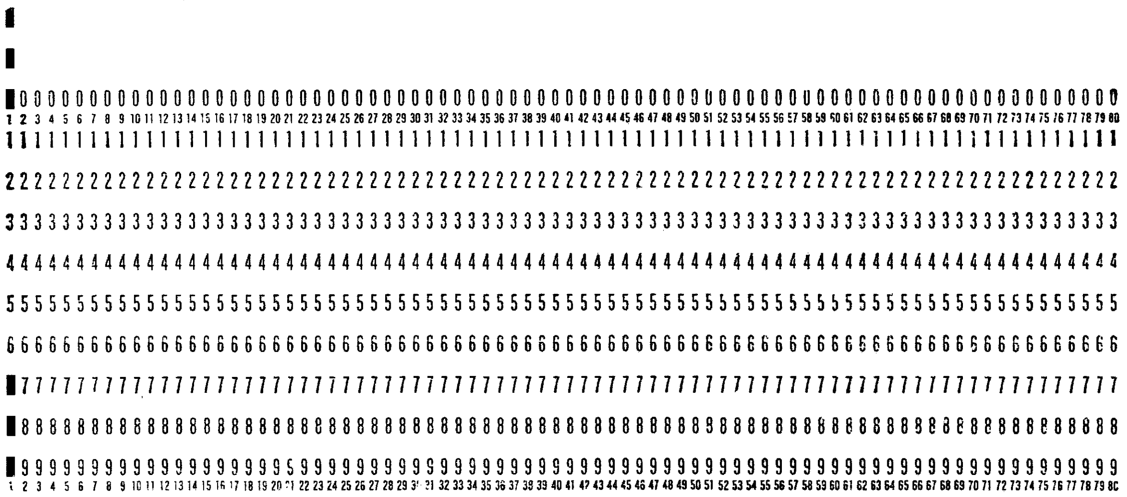


Figure B-4 Blank Suppress Card

B.4 Ø26/Ø29 MODE CARDS

Figures B-5 and B-6 illustrate the Ø26 and Ø29 Mode cards.

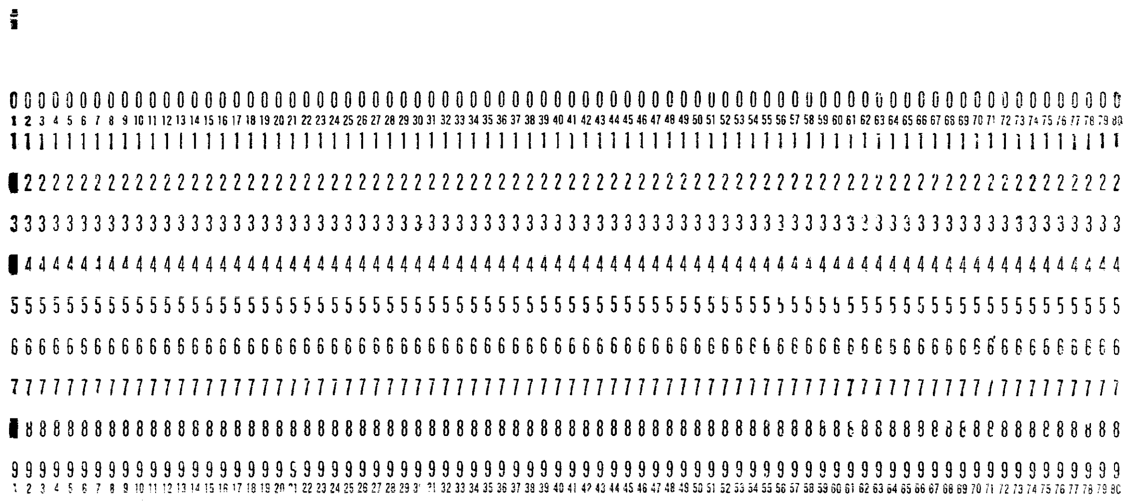


Figure B-5 Ø26 Mode Card (12-2-4-8)

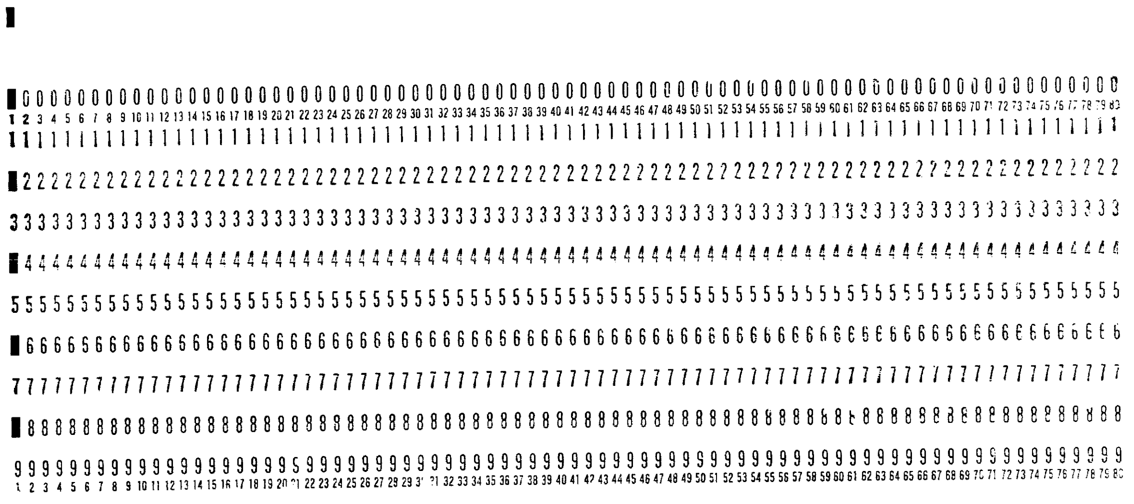


Figure B-6 Ø29 Mode Card (12-Ø-2-4-6-8)

APPENDIX C

OPERATING THE LP11 LINE PRINTER

C.1 CONTROL PANEL

The LP11 Control Panel is located at the top of the cabinet, to the left of the window. This panel contains the controls for normal operation of the printer.

To operate in Batch, the line printers controls and indicator lights must be set as follows:

1. POWER indicator on;
2. READY indicator on;
3. ONLINE/OFFLINE switch in UP (ONLINE) position; and,
4. ONLINE indicator lit. (If the printer is not on-line, the system hangs until the ONLINE/OFFLINE switch is put to ONLINE.)

C.2 MAINTENANCE PANEL

The maintenance panel contains controls used for the line printer's initial set-up and maintenance. It is accessible only by opening the front cabinet door, located beneath the control panel.

This panel contains three switches, and three indicators.

1. Main AC power switch;
2. PRINT INHIBIT switch - must be off (down) to enable printing;
3. DRUM GATE indicator - if lit, drum gate not properly locked;
4. PAPER FAULT - if lit, check for no paper, or torn paper;
5. PRINT INHIBIT indicator - if lit, turn PRINT INHIBIT switch off.
6. MASTER CLEAR switch - spring-loaded to off (down); if toggled to on (up), resets printer logic, turns off READY and ONLINE indicators.

C.3 ADJUSTMENT CONTROLS

Controls are provided as listed in Table C-1.

Table C-1 Adjustment Controls

Control	Location	Function
Drum gate latch	Gearshift type knob near right-hand side of maintenance panel.	Unlocks drum gate which can then be swung open for access to components on back.
Tractor paper width adjustment	Setscrew at far right of tractor pressure plate behind drum gate.	Adjusts right tractor for various paper widths; left tractor is factory adjusted.
Tractor horizontal tension adjustment	Next to left side of tractor paper width adjustment.	Adjusts horizontal tension of paper.
COPIES CONTROL lever	Extreme upper right-hand corner of cabinet just above drum gate hinge.	Adjusts the distance between hammer bank and character drum for different numbers of printed copies. Settings are: 1-2, 3-4 and 5-6.
Paper vertical adjustment control	Knob at upper left of cabinet,directly above right-hand side of maintenance panel	Adjusts vertical alignment of printing so that it prints on lined paper. Can be adjusted to plus or minus one line and may be adjusted while the printer is in operation.
Top-of-form indicators	Red arrows visible when drum gate is swung open one on each side of paper directly below tractor pressure plates	Aligns paper during loading.

C.4 LOADING PAPER

Follow the steps listed below to load paper into the printer.

Step	Procedure
1.	Open front door of cabinet to gain access to maintenance panel and turn main AC power switch on. Verify that control panel POWER indicator lights.
2.	Lift control panel TOP OF FORM switch and release to move tractors to correct loading position.
3.	Open the drum gate by moving the drum gate latch knob to the left and up. Swing drum gate open.

Step	Procedure
4.	Adjust right-hand tractor paper width adjustment for proper paper width. This is accomplished by loosening the set screw on the 8Ø-column model or by using the easy release mechanism on the 12Ø-column model. Make certain that the right-hand tractor is tightened in place after it is adjusted.
5.	Open spring-loaded pressure plates on both tractors.
6.	Load paper so that a perforation is pointed to by the two red arrows (top-of-form indicators). Paper should lie smoothly between tractors without wrinkling or tearing the feed holes.
7.	Close spring-loaded pressure plates on both tractors.
8.	Adjust the COPIES CONTROL lever to the proper number for the number of copies to be made. For example, set to 1-2 for single forms, set to 5-6 for six-part forms.
9.	Close drum gate and lock into position with drum gate latch. After approximately 1Ø seconds the control panel READY indicator should light. If it does not, check to see if any error is indicated. An error is indicated if one of the following lights is on: DRUM GATE, PAPER FAULT, or PRINT INHIBIT.
10.	Lift TOP OF FORM switch several times to ensure paper is feeding properly.
11.	Set system to on-line mode by lifting ON LINE/OFFLINE switch and verifying that ON LINE indicator lights. At this point, printed matter can be aligned with the paper lines by rotating the paper vertical adjustment knob.

For further details on the LP11, refer to the LP11 Line Printer Manual, DEC-11-HLPA-D.

APPENDIX D

FORTRAN LOGICAL DEVICE ASSIGNMENTS

Logical Unit Number	Device	
1	SY:	System device
2	SY:	
3	SY:	
4	PR:	High speed paper tape reader
5	LP:	Line printer
6	KB:	Teleprinter
7	SY:	
8	BI:	Batch input device
3	KB:	(For error logging Dataset name CMO)

Default assignments can be overridden by the $ASSIGN statement. E.g.,

```
$ASSIGN  BI:,3
```

assigns the Batch input device to logical unit 3. A READ to BI would be specified

```
READ(3,...)
```

APPENDIX E

PHYSICAL DEVICE NAMES

Mnemonic	Device	Radix-50 Equivalence
DC	RC11 Disk	014570
DF	RF11 Disk	014760
DK	RK11 Disk	015270
DT	DECtape (TC11)	016040
KB	Teleprinter	042420
LP	Line Printer (LP11)	046600
MT	Magtape (TM11)	052140
PP	High-Speed Paper Tape Punch	063200
PR	High-Speed Paper Tape Reader	063320
CR	Card Reader (CR11)	012620

NOTE

a. Device mnemonics may be three letters on a particular system. The third letter is assigned if there is more than one controller, e.g.:

   DTA for DECtape controller "A"

   DTB for DECtape controller "B"

b. The device name may be followed by an octal number to identify a particular unit when the controller has several device units associated with it, e.g.:

   DT1 indicates unit 1 under a single DECtape control.

   DTA1 indicates unit 1 under controller A in a multicontrol situation.

INDEX

HOW TO OBTAIN SOFTWARE INFORMATION

Announcements for new and revised software, as well as programming notes, software problems, and documentation corrections are published by Software Information Service in the following newsletters.

Digital Software News for the PDP-8 & PDP-12
Digital Software News for the PDP-11
Digital Software News for the PDP-9/15 Family

These newsletters contain information applicable to software available from Digital's Program Library, Articles in Digital Software News update the cumulative Software Performance Summary which is contained in each basic kit of system software for new computers. To assure that the monthly Digital Software News is sent to the appropriate software contact at your installation, please check with the Software Specialist or Sales Engineer at your nearest Digital office.

Questions or problems concerning Digital's Software should be reported to the Software Specialist. In cases where no Software Specialist is available, please send a Software Performance Report form with details of the problem to:

Software Information Service
Digital Equipment Corporation
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

These forms which are provided in the software kit should be fully filled out and accompanied by teletype output as well as listings or tapes of the user program to facilitate a complete investigation. An answer will be sent to the individual and appropriate topics of general interest will be printed in the newsletter.

Orders for new and revised software and manuals, additional Software Performance Report forms, and software price lists should be directed to the nearest Digital Field office or representative. U.S.A. customers may order directly from the Program Library in Maynard. When ordering, include the code number and a brief description of the software requested.

Digital Equipment Computer Users Society (DECUS) maintains a user library and publishes a catalog of programs as well as the DECUSCOPE magazine for its members and non-members who request it. For further information please write to:

DECUS
Digital Equipment Corporation
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

BATCH USER'S GUIDE
DEC-11-OBUDA-A-D
MAY 1972

READER'S COMMENTS

Digital Equipment Corporation maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback -- your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability.

Did you find errors in this manual? If so, specify by page.

How can this manual be improved?

Other comments?

Please state your position. ______ Date: ______

Name: ______ Organization: ______

Street: ______ Department: ______

City: ______ State: ______ Zip or Country ______

- - - - - - - - - - - - - - - - - - Fold Here -

- - - - - - - - - - - - - - - Do Not Tear - Fold Here and Staple - - - - - - - - - - - - - -

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Digital Equipment Corporation
Software Information Services
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

digital equipment corporation